SUPER COMMODORE

Anno 2
n. 5 - Maggio 1985
L. 7.500
Spedizione in abbonamento postale Gruppo III/70

SUPERCOMMODORE la rivista per gli utenti dei prodotti COMMODORE

Un pubblicazione della J.soft EDITRICE
in collaborazione con
GRUPPO EDITORIALE JACKSON



IL MONITOR DEL C16 - PLUS/4

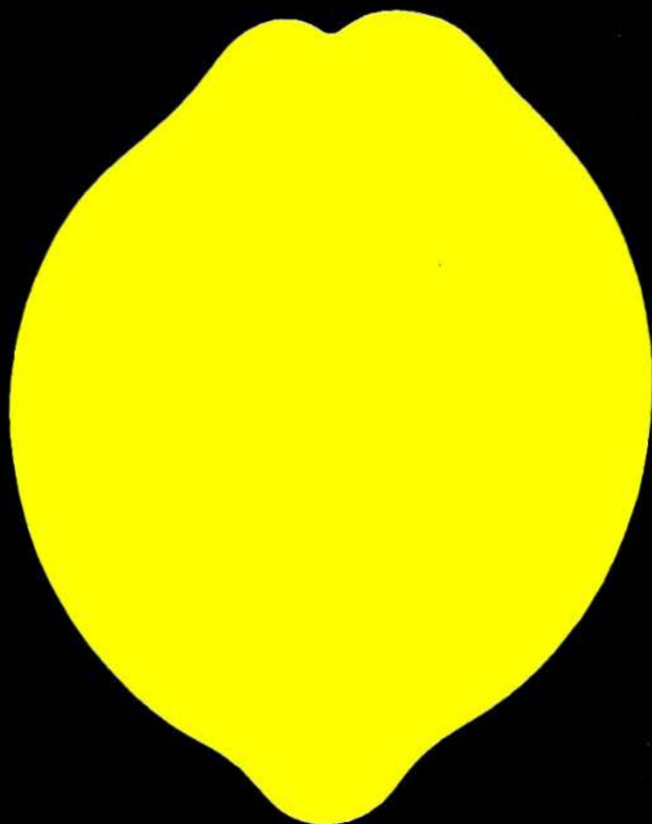
RESCUE OF BLONDELL

KABLAM

PRIGIONE MATEMATICA

ACCORDATORE PER CHITARRA

Per la sete di soft



Il nuovo Lemon II è ancora più potente: nuove soluzioni tecniche e il drive da 160K incorporato consentono a questa macchina prestazioni eccezionali. È possibile infatti utilizzare tutti i migliori programmi professionali: package di gestione, data base, foglio elettronico, word processing, grafica, archiviazione...
Lemon II è compatibile, ha un prezzo accessibile, garanzia e rete di assistenza tecnica.
È l'occasione per avere subito un vero computer professionale.

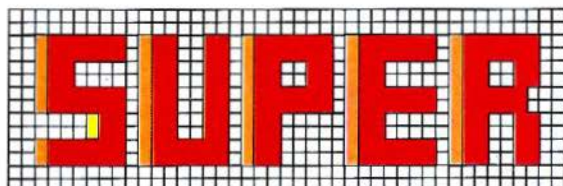


Organizzazione di vendita:
Torino 011.337744
Milano 02.4232437
Bologna 051.223714
Firenze 055.372281
Roma 06.5420305/5423716
Caserta 0823.460469
Catania 095.416560

LEMON II
il grande compatibile

Prodotto e distribuito dalla Belton Electronics s.r.l. - Zona Industriale 62010 Montelupone (MC) - Italia - Tel. 0733/586423-24 Telex 600071 JENSPA I - Divisione informatica della Jen Elettronica s.r.l.

Anno 2 - n° 5 - Maggio 1985



COMMODORE

J. soft s.r.l.
DIREZIONE, REDAZIONE, AMMINISTRAZIONE
Viale Restelli, 5
20124 MILANO
Tel. (02) 68.88.228

DIRETTORE RESPONSABILE:
Pietro Dell'Orco

COORDINAMENTO TECNICO:
Riccardo Paolillo

REDAZIONE:
Mauro Cristuib Grizzi

HANNO COLLABORATO A QUESTO NUMERO
U.G. Barzaghi
A. Motta
F. Sarcina

GRAFICA E IMPAGINAZIONE:
Margherita La Noce
Ivana Rossi
Raffaella Toffolatti

FOTOCOMPOSIZIONE:
Graphotek - Via Astesani, 16
Tel. 02/64.80.397
20161 MILANO

CONTABILITÀ:
Giulia Pedrazzini
Flavia Bonaiti

AUTORIZZAZIONE ALLA PUBBLICAZIONE:
Tribunale di Milano n° 201
del 14.04.1984

STAMPA:
Litografia del Sole
Albairate (MI)

PUBBLICITÀ
Concessionaria per l'Italia e l'Estero J. Advertising s.r.l.
Viale Restelli, 5
20124 MILANO
Tel. (02) 68.82.895-68.80.606-68.87.233
Tlx. 316213 REINA I

Concessionaria esclusiva per la DIFFUSIONE in Italia e Estero:
SODIP - Via Zuretti, 25
20125 MILANO

Spedizione in abbonamento postale Gruppo III/70

Prezzo della rivista L. 7.500
Numero arretrato L. 15.000
Abbonamento annuo: L. 82.500 (11 numeri con cassetta);
L. 38.500 (11 numeri senza cassetta);
per l'estero: L. 110.000 (11 numeri con cassetta) - L. 50.000 (11 numeri senza cassetta)

I versamenti devono essere indirizzati a:
J. soft s.r.l.
Viale Restelli, 5 - 20124 MILANO
mediante emissione di assegno bancario, cartolina vaglia o utilizzando il c/c postale n. 19445204.
Per i cambi di indirizzo indicare, oltre al nuovo, anche l'indirizzo precedente ed allegare alla comunicazione l'importo di L. 500, anche in francobolli

GLI ARTICOLI TRADOTTI SONO TRATTI DALLA RIVISTA COMPUTE! O DA COMPUTE! GAZZETTE - COMPUTE! PUBLICATION, INC.
P.O. BOX 5406,
GREENSBORO, NC 27403 - USA

La biblioteca per ragazzi firmata Jackson

[illegible] Spectrum e ZX 81.
Cod. [illegible] Pag. 48 Lire [illegible]

Ian Graham

GIOCHI CON IL COMPUTER

Il libro, parla di giochi con il computer, una volta tanto visti dalla parte del computer, e non dell'utente.
In particolare spiega, in modo semplice ma preciso, come fa il computer a giocare, come fa a produrre immagini e suoni durante il gioco e come fa (in genere!) a vincere.
Cod. 006D Pag. 48 Lire 9.000

Brian Reffin Smith

IMPARIAMO A PROGRAMMARE

Il libro fornisce le conoscenze essenziali per incominciare a programmare in BASIC su di un personal computer.
Cod. 018D Pag. 48 Lire 9.000

PRIMI PASSI IN BASIC

Il libro propone l'apprendimento del linguaggio BASIC tramite lo studio e l'esame dettagliato di programmi già scritti.
Cod. 007D Pag. 48 Lire 9.000

Tony Potter - Ivor Guild

I ROBOT

Dopo una rapida panoramica su che cosa può fare un robot, il libro presenta una serie di robot con funzioni particolari: i robot a braccio, i robot operai, i robot nello spazio, etc.
Affronta poi il problema di come un robot può essere programmato e controllato da un computer, e di come praticamente si realizza un robot.
Cod. 003D Pag. 48 Lire 9.000

Judy Tatchell - Bill Bennett

CONOSCERE IL PERSONAL

Il libro spiega che cosa si può fare con un personal computer, come si usa e come funziona.
Cod. 008D Pag. 48 Lire 9.000

Lynn Miring - Ian Graham

RIVOLUZIONE INFORMATICA

Il volume è rivolto in particolare ai bambini, ma anche a tutti coloro che, presto o tardi, verranno coinvolti dalle nuove tecniche informatiche più come utenti che come operatori.
Cod. 004D Pag. 48 Lire 9.000



GRUPPO EDITORIALE JACKSON

ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON - Divisione Libri - Via Rosellini, 12 - 20124 Milano

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
		Totale	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 3.000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ____________

n° ____________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome ____________

Cognome ____________

Via ____________

Cap. ______ Città ____________ Prov. ______

Data ______ Firma ____________

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A. | | | | | | | | | | | |

ORDINE MINIMO L. 50.000

POINT



Il computer ha affascinato, fin da principio, solo gli uomini. Sono infatti assai poche le esponenti del gentil sesso che si sono avvicinate all'avveniristico "accrocchio elettronico". Anche tra i lettori e gli abbonati alle nostre riviste si contano pochissime fanciulle. Nonostante l'intraprendenza femminile abbia raggiunto da tempo quella che una volta era propria degli uomini, resta decisamente esiguo il numero di coloro che si interessano di informatica. Appare strano che il calcolatore, con la sua poliedricità, non abbia coinvolto positivamente le donne, alle quali normalmente piacciono le cose che possono assumere funzioni o ruoli diversi.

Forse è la diffusa credenza che il computer sia una fredda macchina, che necessita, per il corretto funzionamento, di studi complicati o di chissà quali alchimie: forse è considerato, a sproposito, un attrezzo riservato a pochi eletti.

Qualunque sia il motivo, resta il dato di fatto che la maggior parte delle donne non ha fatto proprio il dilagante fenomeno che sta rivoluzionando il modo di imparare, di divertirsi e di lavorare insito nella diffusione del personal computer.

Sta dunque a noi convincere le reticenti donzelle a provare almeno una volta il calcolatore, nella speranza che nasca quella curiosità che di solito si trasforma in passione più o meno accesa.

Ragazze, fidanzate, mogli, madri, zie e nonne, nessuna deve sfuggire!

Obblighiamole davanti al video, mostriamo loro quanto sia semplice giocare o scrivere con il computer: se dotate di un minimo di sensibilità non potranno fare a meno di apprezzare i vantaggi offerti dall'uso della macchina infernale; i casi, come sempre, possono essere due: amore a prima vista oppure odio eterno.

Vale comunque la pena di tentare e, se la cosa si risolve positivamente, avrete trovato la tanto attesa giustificazione alle ore di sonno, sempre rinfacciate, perse in compagnia del fedele personal.

Pietro Dell'Orco

Per collaborare a SUPERCOMMODORE

La rivista è interessata ad articoli e programmi riguardanti la linea Commodore, di interesse generale, chiari ed esaurienti. Vi proponiamo una piccola "guida", che ha particolarmente lo scopo di rendere più accurata la stesura dei vostri manoscritti: seguendo i nostri consigli si accresceranno le probabilità che le vostre buone idee e i vostri programmi vengano pubblicati.

1 l'angolo superiore sinistro della prima pagina dovrà contenere: nome, cognome, indirizzo, numero telefonico, codice fiscale, data di spedizione, luogo e data di nascita.

2 l'angolo superiore destro della prima pagina dovrà contenere la marca e il tipo di computer al quale il lavoro si riferisce, unitamente ad eventuali espansioni di memoria o periferiche richieste.

3 il titolo dell'articolo, sottolineato, dovrà iniziare a circa due terzi in altezza della prima pagina.

4 le pagine seguenti potranno essere battute normalmente, con la condizione che l'angolo superiore destro contenga un'abbreviazione del titolo e del cognome, unitamente al numero di pagina. Per esempio, Sprite Ed.../Brambilla/2.

5 il testo dell'articolo dovrà essere battuto a macchina con interlinea di uno spazio e mezzo, massimo due spazi; un margine di almeno un centimetro dovrà essere lasciato su entrambi i lati dello scritto.

6 dovranno essere usati fogli in formato UNI A4 (cm. 21 x 29,7) e il testo, scritto in caratteri maiuscoli e minuscoli, dovrà occupare una sola facciata del foglio.

7 nel caso il testo comprenda più fogli, questi dovranno essere uniti con un fermaglio o con un punto metallico aperto.

8 avendo intenzione di spedire più di un articolo, questi dovranno essere inviati separatamente insieme alla rispettiva copia su supporto magnetico.

9 programmi brevi (meno di 20 linee) potranno essere inseriti nel testo, mentre programmi più lunghi dovranno essere listati separatamente.
È **ESSENZIALE** per noi disporre di una copia del programma registrata più volte su supporto magnetico, su entrambi i lati dello stesso. È preferibile usare nastri di buona qualità e di lunghezza non eccessiva; la cassetta o il disco dovranno essere etichettati con il nome dell'autore, il titolo dell'articolo, il computer interessato e soprattutto le eventuali espansioni richieste. Come suggerimenti di programmazione si consiglia di usare le istruzioni "CHR$(x)", "TAB(x)", SPC(x)", piuttosto che stringhe di manipolazione del cursore.
Ad esempio, per uno scroll di 5 linee l'istruzione "FORI = 1TO5:PRINT:NEXT". è molto più interpretabile di 5 Q inverse; e, invece di una dozzina di simboli di cursore a destra, perché non usare semplicemente "PRINT SPC (12)"? Un rapido controllo dei programmi per operare queste sostituzioni sarà molto apprezzato da noi e dai lettori.

10 per maggior chiarezza, all'interno dell'articolo è conveniente usare caratteri maiuscoli riferendosi a istruzioni BASIC (esempio RETURN, LIST, RND, PRINT ecc.). Se si desidera evidenziare una parola, è preferibile sottolinearla piuttosto che scriverla in carattere maiuscolo.

11 gli articoli ed i programmi potranno avere qualsiasi lunghezza: da una routine di una sola linea fino a programmi molto complessi.

12 volendo includere diapositive, queste dovranno avere formato 24×36, o 6×6.

13 non prenderemo in considerazione articoli che siano stati sottoposti ad altre case editrici.

14 il materiale non pubblicato non verrà restituito.

15 il compenso per la collaborazione prestata sarà commisurato alla complessità e all'interesse del testo e/o del programma (da un minimo di L. **50.000** a un massimo di L. **300.000**). Il pagamento verrà effettuato in caso di pubblicazione del lavoro.

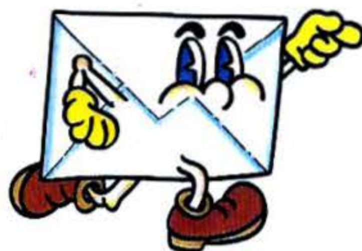
Inviate idee e programmi a:
SUPERCOMMODORE
Viale Restelli, 5
20124 Milano

e saremo lietissimi di pubblicare i contributi migliori.

La Redazione

READ & WRITE

La posta dei lettori



Caricamento automatico del linguaggio macchina

Sono un vostro abbonato alla sola rivista, che ha digitato tutti i programmi in linguaggio macchina per C64 fino ad ora pubblicati. Vorrei chiedervi se esiste qualche sistema per fare in modo che questi ultimi vengano mandati in esecuzione con un semplice RUN invece della SYS, il cui numero si dimentica facilmente.

Giovanni Marchesi

R. Il sistema è molto semplice. Per i programmi in linguaggio macchina che non risiedono nell'area normalmente occupata dai programmi BASIC è sufficiente registrare su nastro o su disco, prima del programma in linguaggio macchina, un cosiddetto "caricatore" BASIC. Questo caricatore – che ovviamente si attiverà con un semplice RUN – si occuperà di caricare il linguaggio macchina e di dare la SYS di attivazione in modo automatico.
Per passare ad un esempio pratico, supponiamo di voler caricare da nastro o disco e mandare automaticamente in esecuzione, senza dover tenere a mente il numero della SYS, il programma "Cosmic Combat" pubblicato sul numero di Febbraio.
Dovremo scrivere questo programma in BASIC:

```
10 IFA=0THENA=1:LOAD"CC",1,1
20 SYS49152
```

Ovviamente occorre modificare il primo 1 in 8, se si utilizza il drive. Questo programma andrà salvato con il nome "Cosmic Combat", mentre il programma in linguaggio macchina vero e proprio dovrà essere salvato subito dopo (questa sequenzialità non è logicamente richiesta se si utilizza il drive) con il nome "CC".
Vediamo ora il perché della insolita IF... THEN come prima istruzione di un programma.
Quando si esegue l'istruzione LOAD da programma, e non in modo diretto, alla fine del caricamento viene forzato un RUN, ed il programma che si troverà in memoria al termine del caricamento verrà quindi mandato in esecuzione. Se il primo programma caricasse un altro programma BASIC, quest'ultimo si sovrapporrebbe al primo e verrebbe automaticamente eseguito una volta caricato; poiché nel nostro caso il secondo programma è il linguaggio macchina, ed occupa un'area di memoria non utilizzata dal BASIC, alla fine del caricamento il RUN automatico avrà ancora effetto sul caricatore, poiché esso si troverà ancora nella memoria BASIC.
Per evitare che venga eseguita all'infinito l'istruzione LOAD c'è appunto questa condizione di IF.... THEN.
Ricapitolando, quando si dà il RUN al caricatore la variabile A vale 0 e viene quindi eseguita l'istruzione LOAD; alla fine del caricamento c'è un RUN automatico, ma la variabile A vale questa volta 1: il LOAD non viene più eseguito ed il programma passa alla linea 20, mandando in esecuzione il programma in linguaggio macchina con l'opportuna SYS.
Chi utilizza il drive invece dell'unità a cassette, e quindi non soffre della perdita dello schermo durante il caricamento, può abbellire il caricatore, scrivendo ad esempio un programma del genere:

```
10 IFA=1THEN60
20 POKE   53280,0:POKE53281,0:
   POKE646,5
30 PRINT"[CLR][3 GIÙ] STO CARI-
   CANDO IL LINGUAGGIO MAC-
   CHINA"
40 PRINT"[2 GIÙ] ATTENDERE PRE-
   GO..."
50 IFA=0THENA=1:LOAD"CC",8,1
60 SYS49152
```

Questa tecnica, lo ricordiamo, non è applicabile a programmi in linguaggio macchina che risiedono nello spazio normalmente occupato dai programmi BASIC (locazione iniziale 2048). Dal momento tuttavia che quasi tutti questi programmi già si attivano con il RUN – essendo dotati di una linea BASIC che effettua la SYS – non avrebbe alcuno scopo utilizzare un caricatore di linguaggio macchina.

Autorun da disco

I programmi registrati su nastro si possono caricare e mandare subito in esecuzione, premendo contemporaneamente SHIFT e RUN/STOP. Dal momento che possiedo solo un disk drive 1541, vorrei sapere se, ed eventualmente come, è possibile ottenere lo stesso risultato caricando i programmi da disco.

Sandro Marchegiani

R. I programmi su disco possono essere caricati e mandati subito in esecuzione digitando LOAD "nome del file", 8: e premendo SHIFT e RUN/STOP quando ci si trova con il cursore dopo i due punti. La pressione di questi due tasti provoca la stampa dell'istruzione LOAD + CHR$(13) – il 13 è il codice ASCII del tasto RETURN – e forza nel buffer di tastiera un'istruzione RUN+CHR$(13). Alla fine del caricamento del programma il buffer di tastiera si svuota, stampando RUN seguito dal codice del RETURN, ed il programma viene quindi attivato.

Piccola guida per l'input dei programmi

Molti dei programmi listati da **SUPERCOMMODORE** contengono particolari caratteri di controllo (controllo cursore, tasti colore, video-inverse, ecc.).
Per rendere chiaro ciò che dev'essere battuto quando è necessario inserire uno dei suddetti caratteri sono state stabilite le seguenti convenzioni.
Generalmente i listati per VIC 20 e C64 contengono delle «parole» racchiuse tra parentesi graffe { }; tali parole rappresentano particolari caratteri di controllo: {**GIÙ**} significa premere il simbolo del cursore verso il basso. {**5 SPAZI**} vuol dire battere 5 volte la barra-spazio.
Per indicare che un tasto dev'essere «shiftato», cioè premuto insieme al tasto SHIFT, nel listato il simbolo di questo tasto sarà sottolineato.
Per esempio S significa che dev'essere premuto il tasto S mentre è abbassato il tasto SHIFT.
La manovra farà apparire sullo schermo un piccolo «cuore».
Trovando un simbolo sottolineato chiuso tra parentesi graffe (esempio {**10 N**}) dovrà essere interpretato come «premere il carattere indicato per il numero di volte che lo precede nelle parentesi»: nel nostro esempio premere 10 volte il tasto N «shiftato».
Se il tasto da premere è indicato tra le parentesi [⟨⟩], vuol dire che lo stesso dovrà essere premuto mentre è premuto il tasto «Commodore» (il tasto «Commodore» è quello posto nell'angolo in basso a sinistra).
Se il tasto scritto tra [⟨⟩] è preceduto da un numero, ciò significa che il tasto dev'essere premuto per il numero di volte indicato.

Raramente si potrà trovare un carattere alfabetico solitario racchiuso tra parentesi graffe. Tale carattere sul C64 può essere battuto mentre è premuto il tasto CTRL.
Ad esempio {**A**} sta ad indicare la sequenza CTRL-A.
A proposito del «modo virgolette» è nota la possibilità di muovere il cursore sullo schermo con i tasti CRSR. Ogni tanto i programmatori desiderano muovere il cursore durante l'esecuzione del programma. È per questo motivo che nei programmi si troveranno dei {**SIN**}, {**HOME**} e {**BLU**}.
L'unico modo perché il calcolatore distingua il comando diretto da quello inserito nel programma è il «modo virgolette». Premendo il tasto «virgolette» (tasti SHIFT 2) il calcolatore si predispone appunto in «modo virgolette».
Battendo un carattere qualsiasi, cercando poi di correggerlo muovendo il cursore, si otterrà solo un tratto verticale in un quadratino inverso. Questo è infatti il simbolo del cursore a sinistra nel «modo virgolette».
L'unico comando di editing non utilizzabile all'interno di un programma è il DEL. Battendo nuovamente le «virgolette» il calcolatore lascia il «modo virgolette». Si può accedere al «modo virgolette» quando inserite spazi nella linea.
In ogni caso, il sistema più semplice per uscire dal «modo virgolette» è quello di premere il tasto RETURN.
Utilizzare la tabella che segue quando è necessario inserire comandi relativi al cursore e ai tasti colore.

Quando leggete:	Premete:	Vedrete:	Quando leggete:	Premete:	Vedrete:	Quando leggete:	Premete:	Vedrete:
{CLR}	SHIFT CLR/HOME		{CYN}	CTRL 4		[<7>]	C= 7	
{HOME}	CLR/HOME	S	{PUR}	CTRL 5		[<8>]	C= 8	
{SU}	SHIFT ⇑ CRSR ⇓		{GRN}	CTRL 6		{F1}	f1	
{GIU'}	⇑ CRSR ⇓	Q	{BLU}	CTRL 7		{F2}	f2	
{SIN}	SHIFT ⇐ CRSR ⇒		{YEL}	CTRL 8		{F3}	f3	
{DES}	⇐ CRSR ⇒		[<1>]	C= 1		{F4}	f4	
{RVS}	CTRL 9	R	[<2>]	C= 2		{F5}	f5	
{OFF}	CTRL 0		[<3>]	C= 3		{F6}	f6	
{BLK}	CTRL 1		[<4>]	C= 4		{F7}	f7	
{WHT}	CTRL 2	E	[<5>]	C= 5		{F8}	f8	
{RED}	CTRL 3		[<6>]	C= 6				





di **M. Cristuib Grizzi**

MSX: quale futuro?



Anche in Italia è iniziata la comparsa sul mercato dei primi computer MSX. Cerchiamo di analizzare cosa differenzia i computer MSX da tutti gli altri e che cosa comporta per il mercato l'entrata di questo nuovo prodotto.

Ogni prodotto che richieda per il suo funzionamento l'utilizzo di mezzi esterni risente, per ottenere successo sul mercato, della maggior o minor reperibilità e compatibilità di questi ultimi. Un esempio di quanto questa affermazione sia vera lo si può trovare nel mercato dei prodotti audio per automobile di qualche anno fa, quando i pur tecnicamente superiori lettori di cassette "stereo 8" vennero in breve tempo fatti scomparire dalla piazza a causa dell'affermazione dello standard "compact cassette". Quest'ultimo venne infatti adottato da un numero sempre maggiore di case produttrici di apparecchi HI-FI per l'auto o per la casa, mettendo il possessore di un apparecchio "stereo 8" in una situazione di sempre più difficile reperibilità di nuove cassette.
L'affermazione sul mercato di uno standard piuttosto di un altro non è solo funzione del livello tecnologico raggiunto dalle varie proposte, ma dipende fortemente dagli interessi economici e dai capitali investiti da parte delle grandi aziende produttrici scese nella competizione. Anche oggi stiamo assistendo alla "guerra" per l'affermazione di uno standard – e quindi di una supremazia – nel settore dei videoregistratori: già ora recenti acquirenti di alcuni di questi prodotti si ritrovano in mano apparecchiature obsolete, non perché superate tecnicamente, bensì perché la maggiore affermazione commerciale di altre marche, espandendo le proprie vendite di cassette ed apparecchiature, comprime fatalmente quelle della concorrenza, costringendo queste aziende ad uscire dal mercato od a riconvertirsi sul breve termine al nuovo standard.
Il mercato degli "home computer", sviluppatosi nel nostro Paese da poco tempo, è sempre stato caratterizzato dai problemi tipici dell'assenza di uno standard. Molte piccole "tragedie personali" si sono consumate sul capo di acquirenti poco informati che, portato a casa dal negozio il nuovo computer di una marca poco diffusa, si sono trovati davanti ad un video vuoto, per la quasi totale irreperibilità sul mercato di software originale o compatibile.
Un esempio di tentativo di standardizzazione nel settore dei personal computer è rappresentato dal CP/M (Control Program for Microcomputers), che, fino a poco tempo fa, era il sistema operativo più diffuso per gli utilizzi gestionali su computer appartenenti alla fascia di

OPEN 

prezzo superiore. Esistono migliaia di programmi scritti sotto CP/M che possono essere eseguiti su una gran quantità di computer differenti, dal portatile al desktop con hard disk. La stessa Commodore ha realizzato nel 1984 una cartuccia per C64 – per altro di difficilissima reperibilità in Italia – che permette di eseguire su questo computer programmi scritti per "girare" sotto CP/M.

A questo punto sembrerebbe ovvio che ogni computer dotato di CP/M possa caricare ed eseguire programmi scritti per "girare" sotto CP/M... Invece è proprio qui che cade il discorso standardizzazione: per fare un esempio, supponiamo che disponiate di un C64 con cartuccia CP/M; come detto, siete in grado di eseguire sul computer qualunque programma scritto per "girare" in ambiente CP/M. Tutto vero, tuttavia non sarete in grado di caricare in memoria da disco il programma, a meno che questo sia stato registrato con un altro C64. Ciò perché la formattazione Commodore dei dischi non è compatibile con altre formattazioni CP/M: così, non potrete semplicemente inserire un disco CP/M nel drive 1541 e caricare un programma CP/M, anche se probabilmente il programma "girerebbe" perfettamente una volta foste riusciti a caricarlo in memoria. Il sogno di poter disporre delle migliaia di programmi CP/M con la semplice inserzione di una cartuccia nel computer crolla così miseramente: dovrete attendere che qualche software house renda il singolo programma disponibile per C64 con cartuccia CP/M.

Lo stesso discorso, passando dal CP/M al MS-DOS e PC-DOS, si può applicare ai computer cosiddetti "compatibili" con il PC IBM i quali non sono mai, a causa dell'aggressività con cui la IBM difende dal punto di vista legale i propri brevetti, compatibili al 100%. È così via...

Sembrerebbe facile, almeno teoricamente, trovare uno standard per gli "home": è sufficiente considerare il computer con il miglior rapporto qualità/prezzo, stabilire quali siano le sue caratteristiche hard-soft fondamentali ed assumerlo come "modello" per tutti i costruttori. In realtà occorre fare i conti con le esigenze di mercato: il computer con il miglior rapporto qualità/prezzo – e per qualità si intende anche una vasta gamma di software disponibile – è in generale quello che ottiene il maggior volume di vendite rispetto alla concorrenza. Per questo motivo ben difficilmente la casa produttrice sarebbe disposta a cederne i diritti di brevetto ad altre aziende, senza porre condizioni economicamente pesantissime e poco concorrenziali per queste ultime.

Il problema va quindi risolto alla base: un gruppo di aziende si deve accordare e progettare "ex novo" un computer che si possa imporre sul mercato per il suo rapporto qualità/prezzo, e la cui politica di commercializzazione ne consenta una grande diffusione, unitamente ad una notevole apertura verso altre aziende desiderose di entrare nello standard.

Lo standard MSX materializza il primo tentativo fino ad ora effettuato per ottenere questi scopi. Fondamentalmente giapponese (Sony, Toshiba, Canon, Pioneer, Yashica, ecc), e con una componente statunitense (Philips e Spectravideo), lo standard MSX è basato su specifiche che devono essere necessariamente possedute da tutti i computer che ne usino il marchio. Queste specifiche si possono riassumere in:

- Microprocessore Zilog Z80A (un microprocessore a otto bit con una frequenza di clock di 4 MHz).
- Memoria ROM di 32 Kbyte, contenente il BASIC MSX ed il Sistema Operativo BIOS.
- Memoria RAM di almeno 8 Kbyte. Una raccomandazione nello standard porta la RAM a 64 Kbyte per il mercato europeo e nordamericano.
- Memoria RAM video di 16 Kbyte, aggiuntiva alla RAM di cui sopra.
- Chip video Texas Instruments TMS9918A, in grado di fornire molteplici modi testo, da uno schermo 24 × 29 ad uno 24 × 40, 256 caratteri ridefinibili a 6 × 8 pixel, una risoluzione massima di 192 × 256 pixel, 16 colori e 32 Sprite, con un massimo di 4 Sprite per linea orizzontale.
- Chip sonoro General Instruments AY-3-8910, a tre voci ed otto ottave, con una risoluzione di frequenza di 12 bit.
- Almeno una porta per espansioni di memoria e software su cartridge, con un'espansione di memoria massima possibile fino al 1 Mbyte.
- Tastiera con un minimo di 70 tasti, controlli cursore separati e cinque tasti funzione.
- Interfaccia per drive a floppy disk MSX-DOS, anche se essa non è completamente standardizzata. Il formato dei dischi è comunque l'MS-DOS, che richiede almeno 64 Kbyte di RAM e permette al drive di leggere dischi formattanti ad esempio su PC IBM o un PC compatibile.
- Interfaccia per unità a cassette con velocità di trasmissione selezionabile tra 1200 e 2400 Baud.
- Dimensioni standard degli slot per cartridge, circuiti standard per l'indirizzamento delle espansioni e standard per tutto ciò che riguarda livelli di segnale sui vari pin, mappa di memoria, punti d'ingresso per il sistema operativo, vettori RAM, ecc.

Queste sono le specifiche *minime* per i computer prodotti con il marchio MSX. Poiché deve comunque esistere una concorrenza anche tra le aziende aderenti allo standard, queste ultime sono lasciate libere di aggiungere ai propri prodotti caratteristiche opzionali che comunque non interferiscano con lo standard (ad esempio video ad 80 colonne, porta RS-232, porta parallela per stampante, ecc.).

La scelta dei componenti effettuata non è casuale: si è cercato di utilizzare una tecnologia che fosse allo stesso tempo relativamente semplice, collaudata ed economica: basti dire, a titolo di esempio, che il chip audio adottato dallo standard MSX è esattamente quello implementato da anni sul TI 99/4A, Coleco Adam e IBM PCjr, mentre il chip video è quello presente sul TI 99/4A e sul Coleco Adam. Il microprocessore Zilog Z80 è da anni implementato sui computer Sinclair, dallo ZX-80 allo Spectrum.

Questo tipo di scelta, pur presentando la massima affidabilità tecnica, rischia di essere troppo conservatrice per una serie di computer nata ora e destinata a resistere per anni sul mercato: si prevede infatti che tra poco più di un anno sarà possibile trovare, a prezzi estremamente interessanti, computer basati su microprocessori a 16 ed anche a 32 bit, dotati di caratteristiche ora disponibili solo su macchine dal costo superiore ai tre milioni. Se ciò avverrà realmente – in un mercato ribollente come questo non si possono che fare previsioni di massima – lo standard MSX diverrà presto obsoleto.

Ritornando al presente, i computer MSX devono già attualmente misurarsi con un'agguerrita concorrenza costituita fondamentalmente da Commodore e Sinclair: un C64 costa poco più della metà di un MSX, dispone di una quantità di software pressoché illimitata e di una grafica con risoluzione superiore. Per contro, il BASIC MSX è un BASIC Microsoft – ricordiamo che MSX sta per MicroSoft eXtended – che può tranquillamente essere definito come il più completo interprete BASIC fino ad ora implementato su qualsiasi personal computer: oltre ad essere velocissimo, dispone di una quantità di istruzioni tale da far letteralmente impallidire al cospetto anche il BASIC Commodore 3.5 implementato di recente sul C16 e Plus/4.

È molto difficile quindi poter prevedere la diffusione che i sistemi MSX avranno nel nostro Paese, anche perché non siamo ora in grado di valutare la portata delle attese novità a 16 e forse anche a 32 bit della Atari e della Commodore.

L'esperimento MSX, qualunque risultato possa sortire, rappresenta un'iniziativa importante che, ci auguriamo, potrà aprire la strada ad una futura e generale standardizzazione dei personal computer.

La piccola guida del principiante

Che cos'è un programma?

Il solo computer non può compiere alcuna operazione.
Un computer possiede potenzialità, ma, come una macchina senza benzina, senza un programma non può funzionare.
La maggior parte dei programmi pubblicati su **SUPERCOMMODORE** per i computer Commodore sono scritti in un linguaggio per calcolatori chiamato BASIC. Il BASIC è facile da imparare ed è disponibile, di serie, nel VIC 20 e nel Commodore 64.

Programmi BASIC

Ogni mese **SUPERCOMMODORE** pubblica programmi sia per il VIC 20 che per il C64. Tanto per cominciare, se possiedi un VIC 20 copia solo i programmi scritti per la tua macchina, indicati con "versione per VIC 20". Più tardi, quando avrai acquisito esperienza con il BASIC del tuo computer, potrai cercare di copiare e convertire programmi scritti per altri calcolatori. Diversamente dal linguaggio corrente, che può essere variamente interpretato, il BASIC di solito ha un solo modo corretto per indicare qualcosa.
Ogni lettera, carattere o numero ha il suo significato.
Un errore banale è costituito dalla sostituzione del numero "0" con la lettera "O" oppure il carattere minuscolo "l" invece del numero "1" o ancora il carattere maiuscolo "B" con il numero "8".
Devi anche inserire tutta la punteggiatura, i due punti (:) e le virgole, copiando esattamente ciò che appare sulla rivista. Gli spazi possono essere importanti. Per essere sicuro **copia il listato esattamente** come si presenta.

Le parentesi e i caratteri speciali

L'eccezione per questa regola di copiatura si presenta quando incontrerai indicazioni tra parentesi, quali: "{GIÙ}".
Ogni cosa compresa tra parentesi è un carattere speciale oppure un carattere che non può essere facilmente prodotto con la stampante. Incontrando un carattere di questo tipo fai riferimento alla "Piccola guida per l'input dei programmi".

Le istruzioni DATA

Alcuni programmi contengono una sezione, o delle sezioni, di istruzioni DATA. Queste linee di istruzione forniscono le informazioni di cui il programma ha bisogno.
In alcuni casi le istruzioni DATA costituiscono il programma vero e proprio, in altri contengono codici grafici. Queste linee sono particolarmente soggette agli errori. Se un solo numero in una linea di istruzione DATA è sbagliato, il calcolatore potrebbe "piantarsi" o distruggere il programma. La tastiera e il tasto STOP appaiono inattivi e lo schermo completamente vuoto. Non lasciarti prendere dal panico. Non si è verificato alcun danno.
Per riprendere il controllo devi spegnere il computer e successivamente riaccenderlo. Ciò cancellerà qualsiasi programma presente in memoria, per cui è sempre necessario fare il SAVE del tuo programma prima di comandare il RUN.
Se il computer si ferma, puoi caricare (LOAD) il programma e cercare l'errore.
A volte, quando il programma viene "lanciato", un'istruzione DATA errata può causare un messaggio di errore.
Il messaggio di errore potrebbe riferirsi alla linea di programma che legge (READ) il contenuto delle istruzioni DATA.

Come conoscere il computer

Dovresti prendere confidenza con il computer prima di procedere alla copiatura del programma.
Impara le istruzioni che si usano per memorizzare e richiamare i programmi da nastro o da disco. Dovrai conservare una copia del tuo programma, se non vorrai copiarlo ogni volta che lo devi usare. Impara ad usare le funzioni di "editing" della tua macchina. Come puoi correggere un errore? Puoi sempre ricopiare la linea e in questo caso devi sapere come procedere. Sapresti come inserire i caratteri in "inverse", i caratteri minuscoli e quelli di controllo?
Tutto ciò è spiegato nel manuale del calcolatore.

Un veloce ripasso

1) Copia il programma una linea alla volta, con ordine. Premi RETURN alla fine di ogni linea. Usa il tasto "DEL" per correggere gli errori.
2) Confronta la linea copiata con quella presente nella rivista. Puoi controllare l'intero programma nel caso in cui si presenti un errore quando esegui il RUN.
3) Accertati di aver inserito le istruzioni tra parentesi graffe con gli appropriati caratteri di controllo (fai riferimento alla "Piccola guida per l'input dei programmi" che trovi in questa stessa rivista).

Siamo spiacenti di non poter rispondere singolarmente alle richieste di informazioni circa i programmi, prodotti o servizi apparsi su SUPERCOMMODORE.

ABC del Linguaggio macchina:
parte seconda

di **U. G. Barzaghi**

Nella seconda puntata del nostro breve itinerario attraverso le lande inesplorate del linguaggio macchina affronteremo due nuovi argomenti, di fondamentale importanza: gli operatori logici ed i formati di indirizzamento del microprocessore 6502.



Come promesso nell'articolo precedente, anche questi concetti saranno illustrati da un programma pratico, realizzato servendosi dell'assemblatore "Supermonitor" comparso sul numero 1 di "SUPERCOMMODORE". In questo caso non si tratterà (come nella prima parte del nostro breve corso di linguaggio macchina, apparsa sul numero precedente) di un banale programma teorico, ma di una comoda routine che potrete incorporare nei vostri programmi, utilizzando le tecniche illustrate nell'articolo, e che consentirà di aggiornare la posizione degli sprite oltre la duecentocinquantacinquesima colonna (in alta risoluzione) dello schermo, evitando quel fastidioso fenomeno di "lampeggio", ben noto a chi abbia familiarità con gli sprite, inevitabile in BASIC.

Operatori logici e sprite

Una delle ragioni che ci ha spinto all'acquisto di un C64 era, senza alcun dubbio, rappresentata dalla possibilità di disporre di otto "sprite". Fin dai tempi in cui la "Atari" introdusse i concetti di "player" e "missile" nella programmazione dei personal computer, tutti i programmatori dediti alla realizzazione di video-giochi e simulazioni non hanno potuto fare a meno di pensare quali fossero le potenzialità offerte da questo sistema.
Non vorremmo, però, che sottovalutaste questa funzione del vostro calcolatore, considerandola alla stregua di un banale "gadget" da sala-giochi. Le possibilità offerte dagli sprite sono molto più vaste. Ad esempio, è possibile utilizzarli, fissi in una determinata posizione, per definire dei simboli matematici (sommatoria, integrale, radice quadrata, ecc.) che per le loro dimensioni sono difficilmente rappresentabili senza far ricorso ad una ridefinizione completa del set di caratteri.
Questa opzione viene, comunque, sfruttata integralmente nella realizzazione di programmi di animazione grafica, il che non significa necessariamente videogame. Nel nostro caso particolare si trattava di arricchire ed esemplificare graficamente un programma che simula l'andamento del flusso di clienti di un autonoleggio, realizzato per un numero monografico sulla simulazione pubblicato dalla rivista "Bit" del Gruppo Editoriale Jackson.
Nel programma la vettura, noleggiata ad un nuovo cliente, viene fatta uscire dal parcheggio dell'autonoleggio fino a scomparire dal lato destro dello schermo. Terminato il periodo di noleggio essa viene fatta rientrare dalla sinistra. Il problema pratico che ci si poneva,

risolto dal programma in linguaggio macchina qui illustrato, consisteva nella attivazione dello sprite in movimento oltre la 255-esima ascissa X senza che lo sprite stesso "lampeggiasse".
Per coloro che non avessero familiarità con l'argomento spiegheremo brevemente il problema. Il C64 consente di attivare 8 sprite o animazioni, la cui forma viene definita da 63 valori numerici contenuti in blocchi fissi di 64 byte (il 64-esimo byte ha quindi solo funzioni di riempimento), contenuti nella stessa pagina di memoria avente una lunghezza di 16 Kbyte "vista" dal chip di gestione della grafica VIC II. Ciò significa, più dettagliatamente, che i dati che definiscono la forma degli sprite e la memoria di schermo devono essere contenuti nello stesso blocco di 16 Kbyte=16384 byte, il che consente, in teoria, di definire la forma di 256 sprite differenti (identificati dai valori numerici compresi tra 0 e 255).
Gli otto sprite disponibili individuano la propria forma "puntando" uno dei 256 blocchi da 64 byte della pagina da 16 Kbyte con il proprio registro. Questi otto registri rappresentano la sola parte del controllo degli sprite, se si esclude la loro forma, che deve risiedere nella stessa "pagina" di memoria della memoria di schermo. Nel caso in cui quest'ultima occupi le locazioni da 1024 a 2024 nella prima "pagina" da 16 Kbyte (cioè le condizioni all'accensione del calcolatore) i registri per gli sprite sono rappresentati dalle locazioni da 2040 a 2047 (2040 indica la forma dello sprite 0, 2047 quella dello sprite 7). Se spostassimo la memoria di schermo al quarto banco di memoria (ad esempio perché desideriamo ridefinire l'intero set di caratteri) a partire dalla locazione 51200, la memoria di schermo terminerebbe in 52200 ed i puntatori alla forma degli sprite occuperebbero gli ultimi otto byte del blocco da 1 Kbyte che contiene la memoria di schermo, vale a dire da 52216 a 52223.
Oltre a questa parte "rilocabile", cioè trasferibile insieme alla memoria di schermo, dei registri degli sprite, abbiamo una serie di registri che ne definiscono altri aspetti e che risiedono in locazioni fisse della memoria controllate dal chip VIC II. Sia detto per inciso, bisogna fare molta attenzione quando si inseriscono in memoria i valori numerici che definiscono la forma degli sprite a non interferire con queste ed altre locazioni "fisse" riservate al sistema operativo del calcolatore. Ciò significa che dei 256 possibili blocchi di 64 byte che compongono un pagina non tutti saranno sempre disponibili per ospitare le nostre animazioni; si renderà perciò necessario un accurato lavoro di pianificazione "sulla carta" della mappa di memoria del nostro Commodore, stabilendo a priori quale blocco di 16 Kbyte si desidera controllare e quali locazioni di memoria siano destinate a uno o più "schermi" (la memoria colore è fissa alle locazioni 55296-56295), quali ad un eventuale set di caratteri personalizzato, memoria a mappa di bit (alta risoluzione) o, appunto, definizione della forma degli sprite.

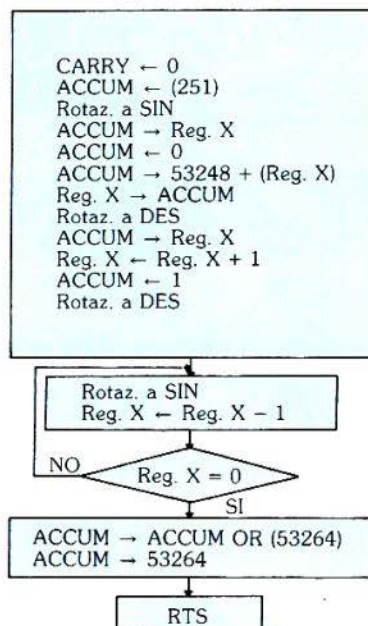


Fig. 1 – How chart

I registri di controllo degli sprite occupano le locazioni da 53248 a 53294 del chip VIC II, ma ad esse sono mischiati altri registri (come i puntatori alla mappa di memoria e i registri di richiesta delle interruzioni – Interrupt ReQuest, IRQ – e la maschera del registro delle interruzioni – Interrupt Request Mask, IRM –) che nulla hanno a che fare con le animazioni.
I primi 16 registri contengono le coordinate degli sprite: la locazione 53248 contiene l' ascissa X dello sprite 0 e la locazione seguente l'ordinata Y dello sprite 7, e così via fino alle 53262-53263, che contengono le coordinate dello stesso sprite. Come abbiamo visto nel numero precedente, la parola di memoria, cioè il numero di bit che compongono una locazione di memoria (di cui il microprocesore 6502 dispone), è composta da 8 bit, il che consente di definire valori numerici compresi tra 0 e 2↑8−1=255.
Ciò non costituisce un problema per l'ordinata Y degli sprite, dal momento che l'area visibile dello schermo, in senso verticale, si estende dalla cinquantesima alla duecentocinquantesima riga di scansione del video.
Rappresenta, invece, un effettivo inconveniente per quanto riguarda la posizione orizzontale delle animazioni, dal momento che, in questo senso, l'area visibile comprende le colonne di pixel (cioè le singole locazioni identificabili sullo schermo) da 24 a 343.
Poiché non è possibile contenere interamente l'ascissa delle singole animazioni nell'apposita locazione ad essa destinata, si è adottato un semplice espediente. È stato definito un ulteriore registro, in 53264, i cui otto bit stabiliscono se l' ascissa dello sprite corrispondente debba intendersi in senso assoluto o a meno di uno scarto pari a 255 pixel. Tutto ciò sembrerà molto più chiaro con un semplice esempio: supponiamo che l'ascissa dello sprite 0, in 53248, e quella dello sprite 1, in 53250, contengano entrambe il valore numerico 45 e che la configurazione del registro 53264 sia la seguente:

Locazione 53264

Bit	7	6	5	4	3	2	1	0
	?	?	?	?	?	?	0	1

In questo caso all' ascissa dello sprite 0 va aggiunto 255 (il bit corrispondente del registro 53264 è posto a uno), mentre quella dello sprite 1 vale effettivamente 45 (il bit 1 del registro suddetto è nullo). In pratica, lo sprite 0 sarà posto in 255 + 45 = 300, mentre lo sprite 1 si troverà effettivamente in 45.
Il problema del "lampeggio" dello sprite, cui il nostro programma in linguaggio macchina intende ovviare, insorge nel caso in cui lo sprite venga spostato lungo lo schermo ed "attraversi" la frontiera rappresentata dalla 255-esima posizione, sia da sinistra verso destra (x crescenti) che in senso opposto (x decrescenti).
Le operazioni richieste nel primo caso sono le seguenti: attivazione del bit del registro 53264, corrispondente allo sprite che attraversa la frontiera (lasciando invariata la configurazione restante!), ed azzeramento dell' ascissa X dello sprite. In qualsiasi ordine queste operazioni vengano eseguite risultano, in BASIC, troppo lente per impedire la comparsa dell'effetto "lampeggio". Vediamo in dettaglio cosa succede: supponiamo che lo sprite 0 attraversi la frontiera da sinistra verso destra. La sua coordinata X viene incrementata fino a raggiungere il valore 255. A questo punto un ulteriore incremento dell'ascissa, ed un tentativo di inserire il valore 256 nella locazione 53248, provocherebbe una segnalazione di errore; un test che stabilisca il raggiungimento di queste condizioni dovrebbe attivare un sottoprogramma che svolga, per lo sprite in questione, le due operazioni specificate più sopra: per azzerare l'ascissa è sufficiente l'istruzione.

POKE 53248,0

mentre per attivare il bit 0 della locazione 53264, lasciando invariati i bit restanti, è necessario utilizzare una istruzione un po' più complessa

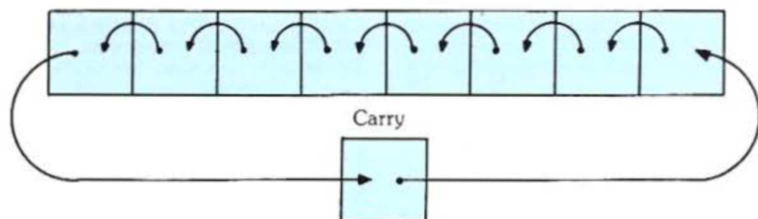


figura 2.a – Istruzione di rotazione a sinistra ROL

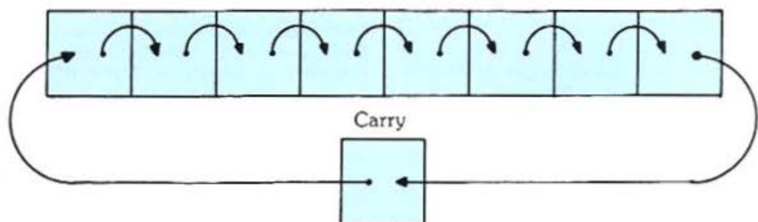


Figura 2.b – Istruzione di rotazione a destra ROR

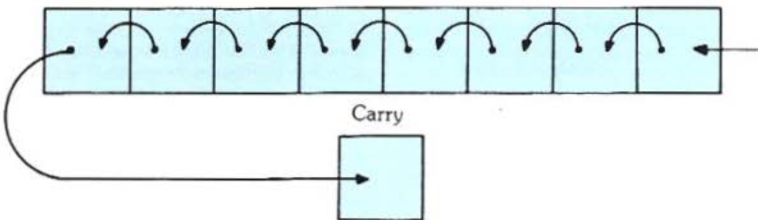


Figura 2.c – Istruzione di scorrimento a sinistra ASL

```
POKE 53264, PEEK (53264) OR1
```

Nel caso di un generico sprite contraddistinto dalla variabile SP (0< = SP< = 7) le due istruzioni suddette assumono la forma

```
POKE 53248+SP*2,0
POKE 53264,PEEK(53264)OR2↑SP)
```

Il problema è rappresentato dall'intervallo di tempo tra la prima e la seconda istruzione. Nell'ordine indicato l'ascissa viene azzerata e, non essendo ancora attivato il bit opportuno della 53264, lo sprite "scompare" dallo schermo (non è compreso nell'area visibile); l'attivazione del bit fa "ricomparire" lo sprite, provocando il lampeggio. Invertendo l'ordine dei fattori il prodotto non cambia... Attivando per primo il bit della locazione 53264 lo sprite ha un'ascissa pari al valore contenuto nella locazione destinata allo scopo (nel nostro caso 255), più duecentocinquantacinque, uscendo, ancora una volta, dal campo visibile o, nel caso in cui si incrementi l'ascissa per valori maggiori di uno, "saltando" nel migliore dei casi in una posizione visibile, ma 255 pixel più a destra lungo lo schermo. Azzerando quindi l'ascissa si riporta lo sprite nella posizione desiderata, ma si ottiene ancora una volta il lampeggio.
Che questo fastidioso effetto sia risolvibile con un semplice aumento della velocità di esecuzione delle due operazioni è facilmente dimostrabile. Come è noto, l'occhio umano non è in grado di distinguere due immagini che si susseguono sulla retina in un tempo inferiore al decimo di secondo (la qual cosa ha presumibilmente a che fare con la velocità di composizione delle immagini del nostro apparato visivo); su questo semplice principio si basano la cinematografia, i cartoni animati e la televisione. Se il vostro occhio fosse piu "veloce", guardando lo schermo del monitor del vostro calcolatore potreste vedere effettivamente il pennello elettronico (cioè il fascio di elettroni del tubo catodico) che compone l'immagine sullo schermo (come è possibile notare in una ripresa, non sincronizzata, di un apparecchio televisivo da parte di una telecamera). L'impiego del linguaggio macchina consente, quindi, di risolvere il nostro problema. Esamineremo in dettaglio l'attraversamento della frontiera a "quota 255" da sinistra a destra, quindi modificheremo il programma, in modo che possa essere utilizzato anche per lo spostamento in senso inverso. Prima, però, dobbiamo approfondire le nostre conoscenze sugli operatori logici di cui il 6502 dispone.

And, Or e Eor

Probabilmente conoscete già almeno due degli operatori logici posseduti dal microprocessore 6502, dal momento che essi sono disponibili anche in BASIC: AND e OR.
L'operatore AND, o prodotto logico, può essere paragonato al collegamento in serie di due interruttori. È sufficiente che l'uno o l'altro di essi sia disinserito perché l'uscita sia nulla, cioè la "lampadina" rimanga spenta.
Trasferendo il ragionamento dall'elettrotecnica all'informatica, è sufficiente che uno qualsiasi dei due bit cui l'operatore viene applicato sia nullo perché il risultato del prodotto logico dei due bit sia nullo. Questo discorso può essere riassunto nella seguente "tabella della verità"

		primo bit	
		0	1
secondo bit	0	0	0
	1	0	1

Applicando l'operatore logico AND a due locazioni si ottiene il prodotto logico del loro contenuto. Ad esempio

Bit	7	6	5	4	3	2	1	0
Loc. A	1	0	0	1	0	1	1	0
Loc. B	1	1	0	0	1	1	0	1
A AND B	1	0	0	0	0	1	0	0

Nel caso del microprocessore 6502 una delle due locazioni coinvolte nella operazione di prodotto logico sarà sempre rappresentata dall'accumulatore, mentre è possibile specificare, nei formati che incontreremo nel proseguo dell'articolo, il secondo elemento dell'operazione.
L'operatore OR, o somma logica, può essere paragonato, tanto per mantenere l'analogia elettrotecnica, al collegamento in parallelo di due interruttori (per avere l'accensione della lampadina è sufficiente che uno o l'altro, o entrambi gli interruttori siano inseriti) ed ha la tabella della verità seguente

		primo bit	
		0	1
secondo bit	0	0	1
	1	1	1

Applicando l'operatore OR alle stesse due locazioni del caso precedente si otterrebbe quindi

Bit	7	6	5	4	3	2	1	0
Loc. A	1	0	0	1	0	1	1	0
Loc. B	1	1	0	0	1	1	0	1
A OR B	1	1	0	1	1	1	1	1

Anche in questo caso l'operatore OR (o, come si dovrebbe specificare, OR inclusivo) prevede, nella forma fornita

dal 6502 – ORA –, che uno dei due operandi sia rappresentato dal contenuto dell'accumulatore.
Per descrivere il funzionamento dell'operatore logico EOR, o OR esclusivo, di cui dispone il microprocessore su cui il C64 si basa, l'unica analogia che ci sovviene è la costruzione delle frasi negative di certe lingue (ad esempio inglese, tedesco, latino e, se non ricordiamo male, anche il greco), nelle quali una doppia negazione afferma. Perché il risultato dell'operazione sia unitario, infatti, è necessario che uno o l'altro dei due operandi, MA NON ENTRAMBI, sia uguale a uno. La tabella della verità corrispondente è la seguente

		primo bit	
		0	1
secondo bit	0	0	1
	1	1	1

ed il risultato dell'operazione di OR esclusivo applicata alle locazioni del nostro esempio sarà quindi il seguente

Bit	7	6	5	4	3	2	1	0
Loc. A	1	0	0	1	0	1	1	0
Loc. B	1	1	0	0	1	1	0	1
A EOR B	0	1	0	1	1	0	1	1

Nel nostro programma non ci serviremo dell'operatore OR esclusivo: abbiamo ritenuto però opportuno menzionarlo per completezza.
Qual è l'utilità degli operatori introdotti? Come abbiamo già visto nell'esempio BASIC di attivazione di un particolare bit della locazione 53264, l'operatore OR (in codice mnemonico ASSEMBLY ORA) consente di settare, cioè porre a uno, un particolare bit senza alterare il contenuto degli altri bit che compongono una determinata locazione: supponiamo che la configurazione della locazione 53264 sia la seguente

Bit	7	6	5	4	3	2	1	0
53264	0	1	1	1	0	1	0	1

e che si debba settare il bit 1, corrispondente allo sprite 1. Per ottenere ciò è sufficiente applicare l'operatore ORA al contenuto della locazione indicata, fornendo come secondo operatore un byte che abbia tutti i bit a zero, tranne il bit 1. Vale a dire:

Bit	7	6	5	4	3	2	1	0
53264	0	1	1	1	0	1	0	1
Maschera	0	0	0	0	0	0	1	0
OR	0	1	1	1	0	1	1	1

Nel caso BASIC il contenuto della locazione 53264 veniva letto con l'istruzione PEEK (53264), cui veniva applicato l'operatore OR. Le funzioni della maschera venivano svolte dalla funzione 2^SP, in cui SP era la variabile che identificava lo sprite; nel nostro caso 2↑1=2, vale a dire il valore numerico decimale corrispondente alla codifica binaria della maschera utilizzata.

Se volessimo settare contemporaneamente i bit 3 e 7 della locazione 53264, dovremmo utilizzare l'istruzione BASIC

```
POKE 53264, PEEK (53264) OR (2↑3+2↑7)
```

sintetizzabile in

```
POKE 53264, PEEK (53264) OR 136
```

Infatti 136 corrisponde alla codifica binaria

Bit	7	6	5	4	3	2	1	0
Peso	128	64	32	16	8	4	2	1
Conf.	1	0	0	0	1	0	0	1

L'operatore AND ha funzioni esattamente opposte. Tramite l'uso di una maschera opportuna, permette infatti di disattivare un determinato bit di una determinata locazione, lasciando intatti gli altri.
Il formato della maschera è intuitivo: essa deve presentare un 1 in ogni posizione, tranne che in quella che si desidera disattivare; essa sarà quindi il complemento a uno della maschera utilizzata per settare quello stesso bit (il complemento a uno si ottiene cambiando ogni 1 di un byte in 0 ed ogni 0 in 1). Per disattivare il bit 3 della locazione 53264 dovremmo quindi applicare l'operatore AND al contenuto della locazione con la maschera

Bit	7	6	5	4	3	2	1	0
Peso	128	64	32	16	8	4	2	1
Conf.	1	1	1	1	0	1	1	1

La codifica decimale corrispondente alla maschera è quindi uguale a 247 o, se SP è uguale a 3, 255-2^SP. L'istruzione BASIC che consente di azzerare il bit corrispondente allo sprite SP è quindi la seguente

```
POKE 53264, PEEK (53264) AND (255-2↑SP)
```

Anche in questo caso, se volessimo disattivare due o più bit contemporaneamente, dovremmo sottrarre da 255 la somma dei loro pesi per ottenere la maschera opportuna.



Per poter realizzare il nostro programma in linguaggio macchina dovremo però occuparci ancora di un elemento: le istruzioni di rotazione e scorrimento.

Rol, Ror e Asl

Le istruzioni di rotazione e scorrimento di cui il microprocessore 6502 dispone sono le tre indicate nel titolo del paragrafo. Le prime due (ROL, ROtate Left, rotazione a sinistra e ROR, ROtate Right, rotazione a destra) consentono di spostare di una posizione nella direzione indicata gli otto bit che compongono il byte cui l'istruzione è applicata (è possibile specificare un indirizzo assoluto, in pagina zero o, per default, far ruotare il contenuto dell'accumulatore), facendo cadere il bit estremo nella locazione normalmente destinata a contenere il riporto (Carry), il cui contenuto viene inserito in corrispondenza del bit lasciato libero dallo scorrimento. In pratica l'operazione ha l'aspetto mostrato in figura 2.a, per quanto riguarda l'istruzione ROL, ed in 2.b per la ROR.
La terza istruzione (ASL, Arithmetic Shift Left, scorrimento aritmetico a sinistra) consente lo spostamento dei primi sette bit di un byte di una posizione verso sinistra, mentre l'ottavo va a cadere nel Carry, ma, contrariamente all'istruzione ROL, il bit meno significativo, lasciato libero dallo scorrimento, viene arbitrariamente rimpiazzato da uno zero (vedi figura 2.c).
Una prima funzione per cui si può pensare di utilizzare le istruzioni suddette è, intuitivamente, rappresentata dalla possibilità di posizionare opportunamente una determinata configurazione di bit, ad esempio quella necessaria per svolgere le funzioni di "maschera" per una istruzione AND o ORA (e, in effetti, ce ne serviremo proprio per posizionare il singolo bit necessario per attivare il bit corrispondente allo sprite prescelto della locazione 53264 o, nel secondo sottoprogramma, per far scorrere in corrispondenza dello stesso bit lo 0 necessario per disattivarlo).
Una seconda e, se vogliamo, più importante funzione, quanto meno più ricca di implicazioni, è rappresentata dall'aspetto aritmetico dello scorrimento verso destra o verso sinistra di tutti i bit di un byte. Vediamo di chiarire con un semplice esempio: supponiamo che l'accumulatore contenga la seguente configurazione binaria

Bit	7	6	5	4	3	2	1	0
Peso	128	64	32	16	8	4	2	1
Conf.	0	0	0	1	0	1	0	1

Il valore decimale corrispondente alla configurazione binaria rappresentata (dato dalla somma dei pesi di ogni singo-

lo bit) è uguale a 21. Se al contenuto attuale dell'accumulatore applichiamo una istruzione ASL, la configurazione che otterremo sarà la seguente:

Bit	7	6	5	4	3	2	1	0
Peso	128	64	32	16	8	4	2	1
Conf.	0	0	1	0	1	0	1	0

in cui tutti i bit settati a uno si sono spostati di una posizione verso sinistra e in corrispondenza del bit meno significativo è stato forzato uno zero, mentre lo zero contenuto nel bit 7, il più significativo, è ora contenuto nella locazione destinata al riporto. La codifica decimale della nuova configurazione binaria è uguale a 42=21*2. Ciò significa che lo scorrimento del contenuto di un determinato byte di una posizione verso sinistra equivale a raddoppiarne il valore. Purtroppo il microprocessore 6502 non dispone di una istruzione di scorrimento verso destra; a questo scopo è necessario utilizzare l'istruzione di rotazione ROR, facendo attenzione al contenuto del Carry (che può essere settato tramite l'istruzione SEC – SEt Carry – e annullato con la CLC – CLear Carry –). In effetti, nel nostro programma ci serviremo delle sole istruzioni di rotazione e di un attento uso della locazione di riporto.
Abbiamo quindi scoperto che se vogliamo moltiplicare o dividere il contenuto del nostro byte per un multiplo di due, è sufficiente operare un numero di scorrimenti verso sinistra (se si desidera moltiplicare) o verso destra (per dividere) pari al logaritmo in base 2 del moltiplicatore o del divisore, vale a dire quel valore numerico a cui è necessario elevare la base due per ottenere il valore del moltiplicatore o divisore stesso. Operando gli scorrimenti passo-passo, e registrando opportunamente il contenuto del bit di riporto, è possibile compiere operazioni aritmetiche più complesse.
Ci serviremo anche di questa nozione per realizzare il nostro programma di utilità pratica.

Come progettare il programma desiderato

La prima decisione che è necessario prendere consiste nello stabilire dove inseriremo la codifica decimale del codice mnemonico ASSEMBLY del nostro programma. Nel nostro caso la locazione più opportuna sembrava essere rappresentata dal blocco da 1 Kbyte al di sotto della memoria di schermo del nostro programma (in 51200). Sfortunatamente la stessa zona di memoria è occupata da parte del codice macchina del programma assemblatore "Supermonitor", che desideravamo utilizzare per la realizzazione e la messa a punto del sottoprogramma voluto.
Ovviamente, una soluzione era rappresentata dalla rilocazione, cioè il riposizionamento del programma in linguaggio macchina, in un'area di memoria non interessata dal monitor ed il suo successivo ritrasferimento nella posizione voluta. Ciò avrebbe potuto però comportare, a priori, la necessità di apportare ogni volta modifiche agli indirizzi contenuti in alcune istruzioni del programma (in realtà il prodotto finito è poi risultato interamente rilocabile; gli indirizzi contenuti nelle istruzioni sono tutti relativi, forniscono cioè lo scarto della locazione indirizzata rispetto a quella contenente l'istruzione, non l'indirizzo assoluto).
Una soluzione relativamente semplice è invece rappresentata dalla rilocazione completa del "Supermonitor" stesso, ottenibile seguendo le istruzioni riportate in calce all'articolo di presentazione del programma.
Ripeteremo la sequenza necessaria per coloro che disponessero della sola copia del programma, ma non dell'articolo originale.
È necessario, innanzi tutto, caricare e mandare in esecuzione il programma stesso (per rilocarlo useremo infatti le opzioni di cui dispone).
Quando appare il cursore (.) digitare

```
.T C000 CFFF 2000
```

(ciò consente di trasferire il codice ASSEMBLY contenuto nelle locazioni da $C000 – 49152 – a $CFFF – 53247 – a partire dalla locazione $2000 – 8192 –)

```
.N 2000 2003 6000 C000 CFFF
.N 2012 2E6D 6000 C000 CFFF
```

(che consentono di rilocare opportunamente tutti gli indirizzi assoluti contenuti nelle due aree delimitate dai primi due indirizzi di ogni istruzione)

```
.N 2FB5 2FFE 6000 C000 CFFF W
```

(per riposizionare gli indirizzi assoluti nella tavola dei vettori di "Supermonitor"). È inoltre necessario modificare direttamente il contenuto delle seguenti locazioni, utilizzando il comando M:

Locazione	**Valore precedente**	**Nuovo valore**
2018	C5	25
202A	C3	23
2322	CF	2F
2392	CC	2C
2649	C5	25
2701	C5	25
28A7	C3	23

Dopo aver completato queste operazioni è possibile salvare la versione rilocata del programma con

```
.S 2000 3000 "SUPERM $2000"
```

o con

```
.S 2000 3000 "SUPERM $2000" 08
```

Il nuovo programma verrà mandato in esecuzione con l'istruzione SYS8192.

Il programma, finalmente...

Possiamo ora (!) passare alla effettiva realizzazione della nostra routine in linguaggio macchina. Una prima operazione è rappresentata dall'interfacciamento tra il sottoprogramma ed il programma BASIC che sfrutta il sottoprogramma stesso. Come abbiamo visto nell'articolo precedente, è possibile "passare" dei parametri, utilizzando come archivio una locazione di memoria convenuta. Mentre nel programma di somma a 8 bit visto in precedenza abbiamo utilizzato un formato di indirizzamento "assoluto" (con un indirizzo, cioè, composto da due parole di otto bit, che consentono di indirizzare l'intera memoria disponibile), sfrutteremo in questo caso un formato di indirizzamento più "breve" e, pertanto, più "veloce". Si rende necessaria, a questo punto, una breve spiegazione dei vari formati di indirizzamento e dei loro effetti sul funzionamento del calcolatore.
Il ciclo di esecuzione di una istruzione di codice ASSEMBLY si può suddividere in tre parti

1 – Prelievo della nuova istruzione (fase di "fetch")
2 – Decodifica dell'istruzione
3 – Esecuzione dell'istruzione

La prima parte consiste nell'invio del contenuto del registro Program Counter (contatore di programma), comprendente l'indirizzo dell'istruzione corrente, sul bus indirizzo e l'invio di un segnale di lettura sul bus di controllo. Questa serie di operazioni consente di ottenere la prima parola di otto bit della nuova istruzione, parola che caratterizza l'istruzione stessa, identificandone anche il formato di indirizzamento.
Le fasi di decodifica e di esecuzione hanno una lunghezza, contrariamente alla fase di fetch, dipendente dal formato di indirizzamento dell'istruzione. Abbiamo già avuto modo di incontrare due tipi di istruzioni: quelle che operano su di un registro convenuto del microprocessore, per le quali non è necessario specificare un indirizzo, il cui codice è contenuto in un solo byte, e quelle che operano su di una generica locazione di memoria, per le quali il codice dell'istruzione deve essere seguito dai due byte dell'indirizzo (suddivisi in byte meno e più significativo). I due formati precedenti prendono il nome, rispettivamente, di indirizzamento implicito ed indirizzamento assoluto.
Non è però indispensabile impiegare due byte per specificare un indirizzo di memoria. Come abbiamo visto, otto bit consentono di rappresentare in formato binario i valori numerici decimali da 0 a 255. Limitando i nostri indirizzi a que-

Figura 3 – Listato del primo programma ASSEMBLY

$C400	CLC		;azzera il riporto
$C401	LDA	$FB	;carica in accumulatore il contenuto della locazione 251
$C403	ROL		;rotaz. a sinistra
$C404	TAX		;trasferisce il contenuto dell'accumulatore nel registro X
$C405	LDA	#$00	;viene azzerato l'accumulatore
$C407	STA	$D000,X	;il contenuto dell'accumulatore viene registrato nella locazione di indirizzo 53248+X
$C40A	TXA		;trasferisce il contenuto del registro X in accumulatore
$C40B	ROR		;rotaz. a destra
$C40C	TAX		
$C40D	INX		;incrementa di uno il contenuto del registro X
$C40E	LDA	#$01	;carica 1 in accumulatore
$C410	ROR		
$C411	ROL		
$C412	DEX		;decrementa di uno il contenuto del registro X
$C413	BNE	$C411	;nel caso in cui il bit Z del registro di flag non sia attivato, si ricicla a partire dalla locazione $C411
$C415	ORA	$D010	;viene eseguita la somma logica dell'accumulatore e dell'indirizzo 53264 ($D010)
$C418	STA	$D010	;il risultato della operazione precedente viene reinserito nella locazione 53264
$C41B	RTS		;si rientra in ambiente BASIC

sto campo di valori è possibile ottenere istruzioni più "brevi" e quindi, come abbiamo detto, più "veloci". Questo genere di indirizzamento prende il nome convenzionale di "pagina zero"; l'indirizzamento utilizzato fa cioè a meno del byte più significativo (che viene assunto, per convenzione, uguale a zero; da qui il nome di pagina zero – per pagina si intende, infatti, la serie di locazioni indirizzabile con una sola parola di memoria–).

L'inconveniente principale presentato da questo formato di indirizzamento è costituito dal relativo "affollamento" della pagina zero, che contiene gran parte delle locazioni utilizzate dal sistema operativo e dall'interprete BASIC (entrambi hanno pressanti esigenze di velocità). Per i nostri scopi, comunque, le poche locazioni disponibili, da 251 a 255 e la solitaria locazione 2, saranno sufficienti.

Supponiamo quindi di utilizzare come locazione di interfacciamento la 251, in cui inseriremo da programma BASIC, con una istruzione POKE, l'indice corrispondente allo sprite (da 0 a 7) di cui si desidera azzerare l'ascissa X e settare il bit corrispondente del registro 53264.

La prima istruzione del nostro programma (il cui listato è visibile in figura 3), ad indirizzamento implicito, consente di azzerare la locazione di riporto (Carry) in vista di successive operazioni di scorrimento e rotazione.

L'istruzione successiva consentirà di caricare il contenuto della locazione prescelta di pagina zero in accumulatore.

Come abbiamo visto nei paragrafi precedenti, è possibile moltiplicare per due il contenuto di un byte, semplicemente facendolo scorrere di una posizione verso sinistra. Lo scarto dell'indirizzo della locazione destinata ad ospitare l'ascissa dello sprite di indice SP, rispetto all'indirizzo di base dei registri degli sprite – 53248 –, è dato, come detto, da SP*2.

L'istruzione ROL consente quindi di ricavare lo scarto suddetto dall'indice dello sprite letto nella istruzione precedente, mentre la successiva TAX trasferisce il contenuto dell'accumulatore nel registro X.

L'istruzione LDA #$00 utilizza il formato di indirizzamento immediato "breve" (contraddistinto dal prefisso #) per caricare in accumulatore il valore necessario ad annullare l'ascissa dello sprite desiderato. Questa funzione viene implementata tramite un ulteriore formato di indirizzamento, chiamato indicizzato da registro, consistente nel sommare ad un indirizzo di base specificato il contenuto di un registro implicitamente indicato. Nel nostro caso si sommerà all'indirizzo di base 53248 ($D000, in esadecimale) il contenuto del registro X (pari a 2*SP, come abbiamo visto).

Queste prime sei istruzioni realizzano il primo dei due obiettivi che ci eravamo proposti: azzerare l'ascissa dello sprite indicato. Vediamo ora come implementare il secondo obiettivo.

Dovremo, innanzitutto, riparare ai "danni" fatti dalle istruzioni precedenti. Il contenuto dell'accumulatore è infatti ora nullo, mentre in X abbiamo il doppio del valore dell'indice. Il nostro obiettivo immediato sarà quindi quello di ricostruire l'indice dello sprite in questione, in modo che possa essere utilizzato dal resto del programma.

L'istruzione TXA trasferisce il contenuto del registro X in accumulatore; la successiva ROR lo divide per due, tramite una rotazione verso destra, riottenendo il valore originale dell'indice. Esso viene ritrasferito nel registro X (che verrà utilizzato come contatore del ciclo di scorrimento dell'accumulatore, mediante il quale verrà posizionato correttamente il bit della maschera utilizzata dalla istruzione di somma logica).

Il contenuto del registro X viene incrementato di 1 tramite l'istruzione INX in formato di indirizzamento implicito, in modo da comprendere anche il caso in cui si desideri attivare il bit 0.

Viene quindi caricata in accumulatore la maschera di partenza, rappresentata da un byte interamente composto da bit nulli, tranne il meno significativo.

L'istruzione ROR fa scorrere verso destra il contenuto dell'accumulatore, in pratica azzerandolo, poiché il bit in posizione 0 cade nella locazione destinata ad ospitare il riporto. Questa istruzione ha lo scopo di neutralizzare la seguente istruzione, simmetrica, di scorrimento verso sinistra che sposterebbe la maschera di una posizione a sinistra rispetto a quella voluta.

Non è possibile ovviare a questo inconveniente semplicemente decrementando di uno il limite superiore del contatore (in pratica eliminando l'istruzione INX), poiché il test di terminazione del ciclo viene effettuato al termine della operazione di scorrimento, che viene quindi eseguita, comunque, almeno una volta; ciò porterebbe ad un errore nel caso dello sprite 0, poiché lo spostamento della maschera verso destra attiverebbe il bit 1, prima dell'interruzione del ciclo.

L'istruzione ROL ripristina la configurazione di partenza (bit meno significativo attivato) facendo ruotare verso sinistra ogni bit e caricando nel bit così liberato il contenuto del Carry (uguale a uno, poiché contenente il bit della maschera fattovi cadere dalla istruzione precedente di scorrimento verso destra).

Il contenuto del registro X viene quindi decrementato di una unità; questa operazione, oltre che sul registro X utilizzato come contatore, agisce su due bit del registro di "flag": il bit di riporto C ed il bit Z. Il primo rimane in pratica nullo (dato che il valore contenuto nel registro X in origine è, al massimo, pari a otto e quindi non vengono mai generati riporti, poiché il ciclo viene interrotto quando il contenuto del registro raggiunge lo zero); per quanto riguarda il secondo, esso viene attivato per segnalare che l'ultima operazione eseguita ha portato ad un azzeramento della locazione coinvolta nell'operazione.

Questo "segnale" viene sfruttato dalla successiva istruzione di diramazione, BNE (Branch if Not Equal, salta alla locazione indicata, se il risultato della operazione precedente è diverso da zero), per riciclare alla istruzione di scorri-

mento verso sinistra ($C411) fino a che l'azzeramento del registro-contatore X segnala il corretto posizionamento della maschera. Come è possibile notare dal listato del programma, questa istruzione utilizza un ulteriore formato di indirizzamento, che prende il nome di indirizzamento relativo. Questo formato può essere utilizzato tutte le volte che la locazione cui il salto è destinato è compresa in un blocco di ampiezza pari ad una pagina (255 locazione di memoria), centrato sulla locazione contenente l'istruzione. I valori numerici compresi tra 0 e 127 rappresentano diramazioni "dirette", cioè a locazioni seguenti quella contenente l'istruzione di salto, mentre i valori numerici da 128 a 255, vale a dire quelli aventi il bit più significativo (bit 7) posto a uno, indicano diramazioni inverse, cioè a locazioni precedenti quella corrente ed in cui l'entità del salto è indicata dalla differenza tra 255 ed il valore numerico indicato. Nel caso della istruzione di locazione $C413 l'indirizzo relativo è contenuto nel byte seguente ($C414) e l'entità del salto (255-252) indica una diramazione negativa alla locazione posta tre posizioni prima della locazione attuale, vale a dire proprio $C411, come indicato dall'istruzione.

Questo tipo di indirizzamento ha a che fare con la rappresentazione dei numeri negativi in "complemento a due", un argomento che non abbiamo ancora avuto modo di affrontare. Nell'articolo del mese scorso abbiamo avuto modo di vedere come otto bit consentano di rappresentare i valori numerici interi positivi compresi tra 0 e 255; non abbiamo però affrontato il problema dei numeri negativi. Per poterli individuare sono disponibili addirittura tre formati in aritmetica binaria. Il più intuitivo consiste nell'utilizzare il bit più significativo come bit di segno, assumendo 0 come "+" e 1 come "−", rappresentando il valore assoluto del numero tramite i restanti sette bit. In questo modo è possibile definire valori numerici compresi tra −127 (11111111) e +127 (01111111); notiamo però che lo zero ha, in questo formato, due rappresentazioni (−0=10000000 e +0=00000000). Questo formato, chiamato "binario con segno", presenta l'antipatico inconveniente di non funzionare correttamente quando si addizionano numeri negativi con numeri positivi o numeri negativi tra loro. Infatti:

```
+7            0 0 0 0 0 1 1 1
−5            1 0 0 0 0 1 0 1
-----------------------------
−12           1 0 0 0 1 1 0 0
```

Bit di segno

```
−7            1 0 0 0 0 1 1 1
−5            1 0 0 0 0 1 0 1
-----------------------------
+12       [1] 0 0 0 0 1 1 0 0
```

Riporto Bit di segno

Per ovviare a questi inconvenienti è stata introdotta la rappresentazione in "complemento a uno", in cui ogni numero negativo è ottenuto complementando il corrispondente numero positivo, cambiandone cioè gli uni in zeri e viceversa. In questo caso −3 è rappresentato da

+3 = 00000011
−3 = 11111100

Anche questo formato, però, presenta degli inconvenienti per quanto riguarda le operazioni di somma algebrica, sia tra numeri negativi che tra negativi e positivi. È però possibile, in questo caso, ovviare ad eventuali errori, sommando, a operazione conclusa, l'eventuale riporto generato dall'operazione. Vediamo un semplice esempio

```
+7            0 0 0 0 0 1 1 1
−5            1 1 1 1 1 0 1 0
-----------------------------
          [1] 0 0 0 0 0 0 0 1
                          [1]
              ---------------
+2            0 0 0 0 0 0 1 0
```

il risultato è quindi corretto. Analogamente

```
−7            1 1 1 1 1 0 0 0
−5            1 1 1 1 1 0 1 0
-----------------------------
          [1] 1 1 1 1 0 0 1 0
                          [1]
              ---------------
−12           1 1 1 1 0 0 1 1
```

Per evitare di dover apportare ad ogni somma algebrica la suddetta correzione è stata introdotto il formato in "complemento a due", ottenuto sommando uno al formato in complemento a uno di tutti i numeri negativi. In questo caso −3 è uguale a

```
+3           00000011
Compl.       11111100+
                    1
             --------
             11111101
```

In questo formato le operazioni si svolgono correttamente con le normali regole dell'aritmetica binaria

```
+7            0 0 0 0 0 1 1 1
−5            1 1 1 1 1 0 1 1
-----------------------------
+2        [1] 0 0 0 0 0 0 1 0

−7            1 1 1 1 1 0 0 1
−5            1 1 1 1 1 0 1 1
-----------------------------
−12       [1] 1 1 1 1 0 1 0 0
```

trascurando il riporto.

La maschera contenuta nell'accumulatore viene quindi sommata "logicamente" al contenuto attuale del registro 53264 ($D010) ed il risultato dell'operazione viene reimmagazzinato nel registro 53264 tramite le istruzioni ORA $D010 e STA $D010, che utilizzano entrambe il formato di indirizzamento assoluto (a due byte).

Il programma, di cui è possibile vedere un semplice schema a blocchi in figura

Figura 4 – Funzionamento "sulla carta" del programma

LABEL	ISTRUZIONE		ACCUMUL.	REG. X			COMMENTI
			76543210	76543210	C	Z	
	CLC		????????	????????	0	?	Azzera il Carry
	LDA	$FB	00000011	????????	0	0	251 → ACCUM
	ROL		00000110	????????	0	0	Rotaz. a sin.
	TAX		00000110	00000110	0	0	ACCUM → X
	LDA	# $00	00000000	00000110	0	1	ACCUM ← #0
	STA	$D000,X	00000000	00000110	0	1	ACCUM → 53248+X
	TXA		00000110	00000110	0	0	X → ACCUM
	ROR		00000011	00000110	0	0	Rotaz. a des.
	TAX		00000011	00000011	0	0	ACCUM → X
	INX		00000011	00000100	0	0	X+1 → X
	LDA	# $01	00000001	00000100	0	0	ACCUM ← #1
	ROR		00000000	00000100	1	1	Rotaz. a des.
CICLO	ROL		00000001	00000100	0	0	Rotaz. a sin.
	DEX		00000001	00000011	0	0	X−1 → X
	BNE	CICLO	00000001	00000011	0	0	Se Z< >1 => CICLO
CICLO	ROL		00000010	00000011	0	0	Rotaz. a sin.
	DEX		00000010	00000010	0	0	X−1 → X
	BNE	CICLO	00000010	00000010	0	0	Se Z< >1 => CICLO
CICLO	ROL		00000100	00000010	0	0	Rotaz. a sin.
	DEX		00000100	00000001	0	0	X−1 → X
	BNE	CICLO	00000100	00000001	0	0	Se Z< >1 => CICLO
CICLO	ROL		00001000	00000001	0	0	Rotaz, a sin.
	DEX		00001000	00000000	0	1	X−1 → X
	BNE	CICLO	00001000	00000000	0	1	Se Z< >1 => CICLO
	ORA	$D010	????1???	00000000	0	0	[(ACCUM) OR (53264)] → ACCUM
	STA	$D010	????1???	00000000	0	0	ACCUM → 53264
	RTS		????1???	00000000	0	0	Fine

1, viene quindi chiuso dall'istruzione di ritorno da sottoprogramma (RTS, ReTurn, from Subroutine).

Il suo funzionamento, per quanto difficile da seguire in una descrizione per iscritto, può sembrare apparentemente semplice; vi assicuriamo, però, che la sua messa a punto è stata ben più laboriosa.

Per abbreviare i tempi di lavoro, e non andare incontro ad inutili insuccessi, è spesso conveniente verificare il funzionamento di un programma "sulla carta" prima della sua effettiva esecuzione, col duplice scopo di verificare che il suo funzionamento sia corretto e che i risultati ottenuti coincidano con quelli voluti.

A questo scopo è necessario disporre di un foglio di carta quadrettata su cui riporteremo, di volta in volta, le configurazioni delle varie locazioni interessate dal programma. Non è indispensabile verificare il funzionamento del programma per ogni possibile serie di valori dei parametri in ingresso, anche se è consigliabile analizzare attentamente determinati casi limite o valori dei parametri di ingresso particolari (come l'indice 0 per il nostro programma). Un esempio del funzionamento "sulla carta" del nostro programma (nel caso in cui il valore dell'indice dello sprite che si desidera controllare, inserito da BASIC in 251, sia uguale a 3) è riportato in figura 4 dove il simbolo "→" indica un trasferimento tra locazione di memoria; "←#n" il caricamento immediato del valore n nel registro a sinistra della freccia; "=>" l'azione da intraprendere al verificarsi della condizione posta alla sinistra del simbolo e "?" che il contenuto del bit corrispondente è ignoto o irrilevante ai fini della nostra trattazione.

Come inserire il programma in memoria

È possibile inserire il programma in memoria, come abbiamo già visto nell'articolo precedente, utilizzando le funzioni messe a nostra disposizione da "Supermonitor".



Una volta caricato il nostro programma assemblatore, ed aver dato un comando NEW per risistemare i puntatori all'inizio della memoria riservata al BASIC, è necessario utilizzare l'istruzione SYS8192 per mandare in esecuzione la versione rilocata di "Supermonitor" (SYS49152, se intendete servirvi della versione originale, ma, in questo caso, dovrete cambiare gli indirizzi a cui inserire il codice ASSEMBLY che compone il sottoprogramma).

Per inserire il codice in linguaggio macchina direttamente in memoria sarà sufficiente battere, di seguito al "prompt" ".", il comando di ASSEMBLER ("A") seguito dall'indirizzo della locazione destinata ad ospitare la prima istruzione in codice mnemonico della nostra routine ed il codice mnemonico stesso

```
.A C400 CLC
```

Una volta premuto RETURN, il calcolatore premetterà al codice mnemonico la sua conversione in esadecimale

```
.A C400 18 CLC
```

quindi farà comparire il comando di ASSEMBLER seguito dalla locazione immediatamente successiva all'ultima occupata; nel nostro caso, essendo la prima istruzione contenuta in un solo byte

```
.A C401
```

è quindi possibile proseguire nell'introduzione dell'intero testo del programma, che compare in figura 3. Una volta terminato e dopo essere usciti dal formato ASSEMBLER, premendo RETURN dopo l'indirizzo della locazione seguente all'ultima istruzione del programma (RTS), è possibile registrare il programma su disco, tramite il comando

```
.S C400 C41C "LM1/SPRITE" 08
```

o su nastro, tralasciando di aggiungere 08 dopo il nome del file.

Coloro che non desiderassero servirsi della versione rilocata del programma devono sostituire alla C iniziale degli indirizzi qualsiasi cifra superiore a 1 o le lettere A o B. In questo modo non si interferirà né con la versione originale del programma né con la memoria di schermo.

Come aggiungere il sottoprogramma ai propri programmi BASIC

È ovviamente possibile disporre della routine realizzata, caricandola direttamente da disco con il comando

```
LOAD "LM1/SPRITE", 8, 1
```

o da nastro con

```
LOAD "LM1/SPRITE", 1, 1
```

e facendo seguire al comando

```
NEW [RETURN]
```

per resettare i puntatori, onde poter caricare quindi il programma destinato ad utilizzare il sottoprogramma in questione.

Esiste però un metodo più semplice, che consente di incorporare la routine direttamente nei listati dei programmi che la utilizzano, anche se questo vi costerà qualcosa in termini di occupazione di memoria per il listato BASIC e di tempo per l'effettivo inserimento del sottoprogramma in memoria.

Il sistema consiste nel convertire le cifre esadecimali, che il calcolatore inserisce tra l'indirizzo della locazione che le contiene ed il codice mnemonico delle istruzioni da noi utilizzate, in cifre decimali, servendosi della funzione di conversione esadecimale. Ricordate che la funzione "tratta" numeri esadecimali composti da quattro cifre, quindi per convertire $AA bisognerà utilizzare il comando

```
.$00AA
```

La serie di cifre così ottenuta va quindi inserita in frasi DATA all'interno del programma che deve utilizzare il sottoprogramma (se il programma contiene già frasi DATA, fate attenzione che il posizionamento relativo delle due serie di dati corrisponda all'ordine in cui vengono effettivamente letti). Questi dati andranno letti ed inseriti ordinatamente in memoria a partire dalla locazione prescelta (dato che il programma è rilocabile, è in realtà possibile spostarlo in qualsiasi zona di memoria disponibile). Questa serie di operazioni può essere svolta dalle righe BASIC seguenti

```
10 FORI = 50176 TO 50203 : READA
   : POKEI, A : NEXT
20 DATA24, 165, 251, 42, 170, 169,
   0, 157, 0, 208, 138, 106, 170, 232,
   169, 1, 106, 42, 202, 208
30 DATA252, 13, 16, 208, 141, 16,
   208, 96
```

Per utilizzare il programma è inoltre necessario inserire in 251 il valore numerico corrispondente all'indice dello sprite che si desidera controllare e quindi comandarne l'esecuzione con una istruzione SYS all'indirizzo a partire dal quale si è inserito il codice ASSEMBLY del programma. Se SP è l'indice in questione (con 0<=SP<=7)

```
*****POKE251, SP : SYS50176
```

Come modificare il programma originale

L'ultima parte dell'articolo si occuperà

delle modifiche che è necessario apportare al programma per utilizzarlo nel caso di attraversamento in senso opposto della frontiera posta in 255.
Analizziamo rapidamente gli obiettivi che è necessario implementare; contemporaneamente ne forniremo anche la traduzione BASIC.
La prima operazione svolta dal sottoprogramma originale consisteva nell'azzeramento dell'ascissa dello sprite considerato. L'operazione simmetrica, che è necessario compiere nella nuova versione, consiste nel porre l'ascissa stessa pari a 255. Non vi sono, come è evidente, differenze di rilievo rispetto al programma originale. In BASIC è sufficiente sostituire 255 allo 0 dell'istruzione precedente, quindi

```
POKE 53248+2*SP,255
```

dove SP rappresenta sempre l'indice dello sprite desiderato.
Analogamente, in linguaggio ASSEMBLY, è sufficiente sostituire nell'istruzione

```
LDA #$00
```

il valore caricato in indirizzamento immediato con #$FF, corrispondente alla codifica esadecimale del decimale 255. L'istruzione risulterà quindi

```
LDA #$FF
```

La seconda parte del programma presenta invece alcune rilevanti differenze rispetto allo schema del precedente e, in particolare, richiede una istruzione in più. Vediamo, in dettaglio, qual è lo scopo che ci si prefigge.
Dopo aver eguagliato a 255 il contenuto dell'ascissa dello sprite SP è necessario disattivare il bit corrispondente del registro 53264. In BASIC l'obiettivo può essere raggiunto tramite l'istruzione

```
POKE53264, PEEK(53264) AND (255−2^SP)
```

dove la parte PEEK(53264) consente di disporre del contenuto attuale del registro e la maschera necessaria per l'operatore logico AND (in modo che i bit non interessati rimangano invariati) viene semplicemente ottenuta completando quella necessaria per settare lo stesso bit tramite una operazione di somma logica.
Ovviamente sarebbe possibile applicare lo stesso principio al nostro programma ASSEMBLY. Per simmetria abbiamo però preferito mantenere gli stessi meccanismi del programma originale, utilizzando le istruzioni di rotazione. Il problema, che rende necessario aggiungere una istruzione, è provocato proprio dalle istruzioni di rotazione. Come abbiamo visto nei paragrafi precedenti, infatti, le istruzioni di rotazione e di scorrimento coinvolgono, oltre agli otto bit dell'accumulatore o della locazione interessata, anche il bit di riporto. Nel caso della istruzione di scorrimento ASL il bit meno significativo viene arbitrariamente rimpiazzato da uno zero. Ciò ci impedirà di usarla per la costruzione della nostra maschera (da utiizzare per l'operatore di prodotto logico

Figura 5 – Listato del secondo programma ASSEMBLY

$C600	CLC		;azzera il riporto
$C601	LDA	$FB	;carica l'indice contenuto nella locazione 251
$C603	ROL		;rotaz. a sinistra
$C604	TAX		trasferisce il contenuto dell'accumulatore nel registro X
$C605	LDA	#$FF	;carica in accumulatore la costante 255
$C607	STA	$D000,X	;registra il contenuto dell'accumulatore nella locazione di indirizzo pari a 53248+X ($D000+X)
$C60A	TXA		;trasferisce il contenuto del registro X in accumulatore
$C60B	ROR		;rotaz. a destra
$C60C	TAX		
$C60D	INX		;il contenuto del registro X viene incrementato di uno
$C60E	LDA	#$FE	;carica in accumulatore la costante 254 (configurazione binaria 11111110)
$C610	SEC		;pone il riporto uguale a uno
$C611	ROR		
$C612	ROL		
$C613	DEX		;il contenuto del registro X viene decrementato di uno
$C614	BNE	$C612	;se il bit Z del registro di flag è diverso da 1, si ricicla alla locazione $C612
$C616	AND	$D010	;operazione di prodotto logico sui contenuti dell'accumulatore e della locazione 53264 ($D010)
$C619	STA	$D010	;il risultato dell'operazione precedente viene reinserito nella locazione 53264
$C61C	RTS		;si rientra in ambiente BASIC



Figura 6 – Esempio di funzionamento "sulla carta" della seconda routine

LABEL	ISTRUZIONE		ACCUMUL.	REG. X			COMMENTI
			76543210	76543210	C	Z	
	CLC	•	????????	????????	0	?	Azzera il riporto
	LDA	$251	00000010	????????	0	0	251 →ACCUM
	ROL		00000100	????????	0	0	Rotaz. a sin.
	TAX		00000100	00000100	0	0	ACCUM → X
	LDA	#$255	11111111	00000100	0	0	ACCUM ← #255
	STA	$D000,X	11111111	00000100	0	0	ACCUM → 53248+X
	TXA		00000100	00000100	0	0	X → ACCUM
	ROR		00000010	00000100	0	0	Rotaz. a des.
	TAX		00000010	00000010	0	0	ACCUM →X
	INX		00000010	00000011	0	0	X+1 → X
	LDA	#254	11111110	00000011	0	0	ACCUM ← #254
	SEC		11111110	00000011	1	0	Il riporto viene posto uguale a 1
	ROR		11111111	00000011	0	0	Rotaz. a des.
CICLO	ROL		11111110	00000011	1	0	Rotaz. a sin.
	DEX		11111110	00000010	1	0	X-1 → X
	BNE	CICLO	11111110	00000010	1	0	Se Z< >1 => CICLO
CICLO	ROL		11111101	00000010	1	0	Rotaz. a sin.
	DEX		11111101	00000001	1	0	X−1 → X
	BNE	CICLO	11111101	00000001	1	0	Se Z< >1 => CICLO
CICLO	ROL		11111011	00000001	1	0	Rotaz. a sin.
	DEX		11111011	00000000	1	1	X−1 → X
	BNE	CICLO	11111011	00000000	1	1	Se Z< >1 => CICLO
	AND	$D010	?????0??	00000000	1	?	[(ACCUM) AND (53264)] → ACCUM
	STA	$D010	?????0??	00000000	1	?	ACCUM → 53264
	RTS		?????0??	00000000	1	?	Fine

AND), dal momento che essa deve essere composta interamente da bit settati a uno, tranne che in corrispondenza della posizione legata allo sprite controllato.

Anche per quanto riguarda le istruzioni di rotazione non possiamo essere sicuri del contenuto del riporto (che viene inserito nel bit più o meno significativo); dovremo quindi assicurarci che esso sia settato prima dell'inizio del ciclo che consente di predisporre la maschera (benché il ciclo che ne controlla la costruzione preveda anche una operazione di decremento del registro X, che non genera mai riporti, l'uno che "esce" dall'accumulatore, e viene forzato nel Carry, non viene annullato, dal momento che l'istruzione DEX non influisce sul riporto).

Per ovviare a questo inconveniente è quindi sufficiente premettere alla seconda istruzione ROR una istruzione di inizializzazione del riporto a uno: SEC, cioè SEt Carry. Il listato completo del nuovo programma, rilocato a partire da $C600 (#50688), compare in figura 5.

Come è possibile notare dalla seconda istruzione, viene utilizzata per l'interfacciamento con il programma BASIC la stessa locazione (251) utilizzata dalla routine precedente. Dato che i due sottoprogrammi sono mutuamente esclusivi (se uno dei due è in funzione, non c'è bisogno dell'altro), non è necessario sprecare un'altra locazione inutilmente. Il nuovo sottoprogramma sarà richiamato in maniera non differente dal precedente:

```
*****POKE251, SP:SYS50688
```

Per quanto riguarda il suo inserimento internamente ad un programma BASIC è necessario modificare, oltre ai valori numerici da inserire in frasi DATA – che lasciamo calcolare ai lettori tramite l'opzione "M" di "Supermonitor" –, i limiti del ciclo di lettura ed inserimento dei dati, che vanno inseriti alle locazioni da 50688 a 50717.

In figura 6, infine, vediamo un esempio di funzionamento "sulla carta" della nostra nuova routine, con un valore dell'indice contenuto nella locazione 251 uguale a 2. Sono state utilizzate le stesse notazioni di figura 4.

OK-List per la perfetta battitura dei listati



Inserite in memoria l'OK-LIST prima di accingervi a battere i listati pubblicati: avrete a disposizione un formidabile mezzo per essere sicuri di aver inserito nel computer ogni carattere in modo assolutamente corretto.
Tutti i nostri listati sono caratterizzati da un numero aggiunto ad ogni linea di programma, ad esempio: rem 123. NON COPIATE QUESTA PARTE DELLA LINEA! Serve unicamente per vostra informazione. L'istruzione REM ha l'unico scopo di renderla inoffensiva, se per errore doveste batterla.
Se, prima di iniziare il lavoro di battitura del listato, caricate in memoria OK-LIST e lo attivate con RUN, per ogni linea che battete verrà visualizzato un numero (detto checksum) nell'angolo superiore sinistro dello schermo.
Confrontate questo numero con quello pubblicato per ogni linea del listato: se non corrisponde, ciò significa che avete commesso un errore nella battitura della linea stessa.
Le ore buttate per cercare e correggere errori di battitura in programmi che non funzionano saranno solo un lontano ricordo!
Due ultime cose di cui tenere conto:
OK-LIST non tiene conto degli spazi: questo per vostra convenienza, poiché questi generalmente sono poco importanti.
Infine, OK-LIST è allocato nel buffer del registratore (locazioni 886-1018), quindi, prima di salvare su cassetta il programma che state battendo, ricordatevi di disabitarlo premendo contemporaneamente RUN/STOP + RESTORE. Potrete poi riattivarlo con SYS 886.
Il sistema più comodo per usare OK-LIST è quello di battere normalmente il programma, quindi listarlo e portarsi con il cursore sul numero di linea più basso che appare sul video; premere quindi RETURN e confrontare il numero di check-sum visualizzato nell'angolo superiore sinistro del video con quello pubblicato per la data riga del programma. Il cursore si porterà automaticamente sulla linea successiva e non dovrete fare altro che premere il checksum seguente.
Data l'area di memoria in cui si trova, OK-LIST non può essere usato per controllare un programma già salvato su cassetta, mentre ciò è possibile se il programma è stato registrato su disco.

OK-LIST versione per VIC 20 e C 64

```
100 PRINT"{CLR}ATTENDERE PREGO..."
105 FORI=886TO1018:READA:CK=CK+A:POKEI,A:
    NEXT
110 IF CK<>17539 THEN PRINT"{GIU'}ERRORE
    NELLE ISTRUZIONI DATA":END
120 SYS886:PRINT"{CLR}{ 2 GIU'}OK-LIST AT
    TIVATO.":NEW
886 DATA 173,036,003,201,150,208
892 DATA 001,096,141,151,003,173
898 DATA 037,003,141,152,003,169
904 DATA 150,141,036,003,169,003
910 DATA 141,037,003,169,000,133
916 DATA 254,096,032,087,241,133
922 DATA 251,134,252,132,253,008
928 DATA 201,013,240,017,201,032
934 DATA 240,005,024,101,254,133
940 DATA 254,165,251,166,252,164
946 DATA 253,040,096,169,013,032
952 DATA 210,255,165,214,141,251
958 DATA 003,206,251,003,169,000
964 DATA 133,216,169,019,032,210
970 DATA 255,169,018,032,210,255
976 DATA 169,058,032,210,255,166
982 DATA 254,169,000,133,254,172
988 DATA 151,003,192,087,208,006
994 DATA 032,205,189,076,235,003
1000 DATA 032,205,221,169,032,032
1006 DATA 210,255,032,210,255,173
1012 DATA 251,003,133,214,076,173
1018 DATA 003
```

MLX per VIC 20 e C64 (versione 2.0 per C64)

di **C. Brannon**
trad. e adatt.
di **M. Cristuib Grizzi** e **F. Stella**

MLX è un programma che permette di inserire listati in linguaggio macchina esenti in modo assoluto da errori e senza la perdita di tempo del dover battere e controllare lunghe sequenze di istruzioni DATA.
Molti dei nostri listati di programmi in linguaggio macchina hanno il formato MLX (li riconoscete dal fatto che sono esclusivamente numerici) e richiedono quindi che MLX sia caricato in memoria ed eseguito prima della battitura del listato.
MLX vi chiederà l'indirizzo della locazione di partenza e quello della locazione finale del programma da caricare. Questi valori sono sempre indicati nell'articolo che accompagna il listato. MLX vi segnala automaticamente gli errori di battitura MENTRE STATE DIGITANDO IL LISTATO e vi chiede di reinserire la linea errata. L'ultimo numero battuto di ogni linea rappresenta il checksum e viene visualizzato in reverse.
Sono inoltre disponibili altri comandi, quali:
SHIFT-N nuovo indirizzo: permette di cambiare l'indirizzo della linea che volete battere ed è utile nel caso si inseriscano i listati in più parti.
SHIFT-D display: lista i dati inseriti tra due indirizzi di inizio e fine.
SHIFT-L load: carica un file da nastro o
SHIFT-S save: salva su nastro o disco un file in formato MLX.
Una volta battuto il listato, e salvatolo tramite MLX, si potrà caricare direttamente il programma con un'istruzione LOAD "nome del file",1,1 per il registratore, oppure LOAD "nome del file",8,1 per l'unità a dischi, salvo diverse indicazioni date negli articoli corrispondenti ai vari programmi. Il programma sarà generalmente mandato in esecuzione con una SYS (indirizzo di partenza).

La versione originaria di MLX per C64 ha subito, dalla prima pubblicazione, diversi ritocchi e miglioramenti, fino alla versione 2.0 qui presentata.
È stato in primo luogo modificato il colore di fondo dello schermo e del bordo, in modo da risultare meno stancante alla vista; sono state ulteriormente perfezionate le routine in linguaggio macchina di salvataggio e caricamento dei programmi; è cambiata la forma del cursore e, cosa più importante, è stato aggiunto un tastierino numerico per gli utenti che si trovano più a loro agio con una diversa disposizione dei tasti.
Oltre a poter usare i soliti tasti numerici, sono stati ridefiniti alcuni tasti alfabetici, in modo da generare ugualmente dei numeri. I tasti ridefiniti sono i seguenti:

U I O		7 8 9
H J K L	diventano	0 4 5 6
M , .		1 2 3

Le persone abituate ad usare tali tastierini accresceranno senz'altro, in brevissimo tempo, la velocità di battitura dei programmi in linguaggio macchina.

MLX versione per VIC 20

```
100 PRINT"{CLR}{PUR}";CHR$(142);CHR$(8);
                                :rem 181
101 POKE788,194:REM DISABILITA RUN/STOP
                                :rem 144
110 PRINT"{RVS}{ 14 SPAZI}"     :rem 117
120 PRINT"{RVS} {DES}{OFF}[<*>]£{RVS}
    {DES} {DES}{ 2 SPAZI}[<*>]{OFF}[<*>]
    £{RVS}£{RVS} "              :rem 191
130 PRINT"{RVS} {DES} [<G>]{DES}
    { 2 DES} {OFF}£{RVS}£[<*>]{OFF}[<*>]
    {RVS} {OFF}"                :rem 122
140 PRINT"{RVS}{ 14 SPAZI}"     :rem 120
200 PRINT"{ 2 GIU'}{PUR}{BLK}UN PROGRAMMA
     PER L'IN-TRODUZIONE{ 2 SPAZI}DI ROUT
    INEIN LINGUAGGIO MACCHINA"; :rem 202
205 PRINT"A PROVA DI ERRORE{ 5 GIU'}"
                                :rem 226
210 PRINT"{BLK}{ 4 SU}INDIRIZZO DI PARTEN
    ZA":INPUTS:F=1-F:C$=CHR$(31+119*F)
                                :rem 52
220 IFS<256ORS>32767THENGOSUB3000:GOTO210
                                :rem 2
225 PRINT:PRINT:PRINT:PRINT     :rem 123
230 PRINT"{BLK}{ 3 SU}INDIRIZZO CONCLUSIV
    O":INPUTE:F=1-F:C$=CHR$(31+119*F)
                                :rem 170
240 IFE<256ORE>32767THENGOSUB3000:GOTO230
                                :rem 234
250 IFE<STHENPRINTC$;"{RVS}INDIRIZZO CONC
    LUSIVO <INDIRIZZO DI PARTENZA
    { 3 GIU'}":GOSUB1000:GOTO230:rem 27
260 PRINT:PRINT:PRINT           :rem 179
300 PRINT"{CLR}";CHR$(14):AD=S  :rem 56
310 PRINTRIGHT$("0000"+MID$(STR$(AD),2),5
    );":";:FORJ=1TO6            :rem 234
320 GOSUB570:IFN=-1THENJ=J+N:GOTO320
                                :rem 228
390 IFN=-211THEN710             :rem 62
400 IFN=-204THEN790             :rem 64
410 IFN=-206THENPRINT:PRINT"{GIU'}INSERIS
    CI UN NUOVO{ 4 SPAZI}INDIRIZZO";:INPU
    TZZ                         :rem 42
415 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"
    {RVS}ESCE DAL CAMPO DI VA-" :rem 150
416 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"
    {RVS}LORI INDICATO":GOSUB1000:GOTO410
                                :rem 105
417 IFN=-206THENAD=ZZ:PRINT:GOTO310
                                :rem 238
420 IFN<>-196THEN480            :rem 133
430 PRINT:INPUT"LISTATO:DA";F:PRINT,"
    { 2 SIN}A";:INPUTT          :rem 29
440 IFF<SORF>EORT<SORT>ETHENPRINT"MINIMO"
    ;S;"{SIN}, MASSIMO";E:GOTO430
                                :rem 36
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHT$("000
    0"+MID$(STR$(I),2),5);":";  :rem 30
455 FORK=0TO5:N=PEEK(I+K):IFK=3THENPRINTS
    PC(10);                     :rem 34
457 PRINTRIGHT$("00"+MID$(STR$(N),2),3);"
    ,";                         :rem 157
460 GETA$:IFA$>""THENPRINT:PRINT:GOTO310
                                :rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRIN
    T:GOTO310                   :rem 50
480 IFN<0THENPRINT:GOTO310      :rem 168
490 A(J)=N:NEXTJ                :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CK
    SUM=(CKSUM+A(I))AND255:NEXT :rem 200
510 PRINTCHR$(18);:GOSUB570:PRINTCHR$(20)
                                :rem 234
515 IFN=CKSUMTHEN530            :rem 255
520 PRINT:PRINT"LA RIGA E' STATA INSE-RIT
    A IN MANIERA ERRATA"        :rem 39
525 PRINT"RI-INSERIRLA":PRINT:GOSUB1000:G
    OTO310                      :rem 167
530 GOSUB2000                   :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT
                                :rem 80
550 AD=AD+6:IFAD<ETHEN310       :rem 212
560 GOTO710                     :rem 108
570 N=0:Z=0                     :rem 88
```

```
580 PRINT"[<+>]";                          :rem  79
581 GETA$:IFA$=""THEN581                   :rem  95
585 PRINTCHR$(20);:A=ASC(A$):IFA=13ORA=44
    ORA=32THEN670                          :rem 229
590 IFA>128THENN=-A:RETURN                 :rem 137
600 IFA<>20THEN630                         :rem  10
610 GOSUB690:IFI=1ANDT=44THENN=-1:PRINT"
    {SIN} {SIN}";:GOTO690                  :rem 172
620 GOTO570                                :rem 109
630 IFA<48ORA>57THEN580                    :rem 105
640 PRINTA$;:N=N*10+A-48                   :rem 106
650 IFN>255THENA=20:GOSUB1000:GOTO600
                                           :rem 229
660 Z=Z+1:IFZ<3THEN580                     :rem  71
670 IFZ=0THENGOSUB1000:GOTO570             :rem 114
680 PRINT",";:RETURN                       :rem 240
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211)
                                           :rem 149
692 FORI=1TO3:T=PEEK(S%-I)                 :rem  68
695 IFT<>44ANDT<>58THENPOKES%-I,32:NEXT
                                           :rem 205
700 PRINTLEFT$("{ 3 SIN}",I-1);:RETURN
                                           :rem   7
710 PRINT"{CLR}{RVS}*** SAVE ***
    { 3 GIU'}"                             :rem 236
720 INPUT"{GIU'} NOME DEL FILE";F$
                                           :rem 199
730 PRINT:PRINT"{ 2 GIU'}{RVS}N{OFF}ASTRO
    O {RVS}D{OFF}ISCO: (N/D)"              :rem 128
740 GETA$:IFA$<>"N"ANDA$<>"D"THEN740
                                           :rem  30
750 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$
                                           :rem 158
760 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256                       :rem   3
762 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                           :rem 109
763 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                                     :rem  69
765 POKE254,S/256:POKE253,S-PEEK(254)*256
    :POKE780,253                           :rem  12
766 POKE782,E/256:POKE781,E-PEEK(782)*256
    :SYS65496                              :rem 124
770 IF(PEEK(783)AND1)OR(ST AND191)THEN780
                                           :rem 111
```

```
775 PRINT"{GIU'}OPERAZIONE CONCLUSA.":END
                                           :rem 152
780 PRINT"{GIU'}ERRORE DI REGISTRAZIO-NE.
    RIPROVA.":IFDV=1THEN720                :rem   8
781 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO720                   :rem 103
782 GOTO720                                :rem 115
790 PRINT"{CLR}{RVS}*** LOAD ***
    { 2 GIU'}"                             :rem 212
800 INPUT"{ 2 GIU'} NOME DEL FILE";F$
                                           :rem 215
810 PRINT:PRINT"{ 2 GIU'}{RVS}N{OFF}ASTRO
    O {RVS}D{OFF}ISCO: (N/D)"              :rem 127
820 GETA$:IFA$<>"N"ANDA$<>"D"THEN820
                                           :rem  28
830 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$
                                           :rem 157
840 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256                       :rem   2
841 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                           :rem 107
845 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                                     :rem  70
850 POKE780,0:SYS65493                     :rem  11
860 IF(PEEK(783)AND1)OR(ST AND191)THEN870
                                           :rem 111
865 PRINT"{GIU'}OPERAZIONE CONCLUSA.":GOT
    O310                                   :rem 142
870 PRINT"{GIU'}ERRORE DI CARICAMENTO.RIP
    ROVA.{GIU'}":IFDV=1THEN800             :rem  43
880 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO800                   :rem 102
1000 REM CICALINO                          :rem 231
1001 POKE36878,15:POKE36874,190            :rem 206
1002 FORW=1TO300:NEXTW                     :rem 117
1003 POKE36878,0:POKE36874,0:RETURN
                                           :rem  74
2000 REM CAMPANELLO                        :rem 130
2001 FORW=15TO0STEP-1:POKE36878,W:POKE368
     76,240:NEXTW                          :rem  22
2002 POKE36876,0:RETURN                    :rem 119
3000 PRINTC$;"{RVS}NON IN PAGINA ZERO O
     { 2 SPAZI}SU ROM{ 4 GIU'}":GOTO1000
                                           :rem  23
```

MLX versione 2.0 per C64

```
100 PRINT"{CLR}{CYN}";CHR$(142);CHR$(8);:
    POKE53280,0:POKE53281,0                :rem  71
101 POKE788,52:REM DISABILITA RUN/STOP
                                           :rem  89
110 PRINT"{RVS}{ 40 SPAZI}";               :rem 176
120 PRINT"{RVS}{ 15 SPAZI}{DES}{OFF}[<*>]
    ~{RVS}{DES} {DES}{ 2 SPAZI}[<*>]{OFF}
    [<*>]~{RVS}~{RVS}{ 13 SPAZI}";
                                           :rem 250
130 PRINT"{RVS}{ 15 SPAZI}{DES}[<N>][<H>]
    {DES} { 2 DES} {OFF}~{RVS}~[<*>]{OFF}
    [<*>]{RVS}{ 13 SPAZI}";                :rem 220
140 PRINT"{RVS}{ 40 SPAZI}"                :rem 120
200 PRINT"{ 2 GIU'}{PUR}EDITOR DI CODICE
    MACCHINA VERSIONE 2.0{ 3 GIU'}"
                                           :rem 126
210 PRINT"[<5>]{ 2 SU} INDIRIZZO DI PARTE
    NZA{ 2 SPAZI}";:INPUTS:F=1-F:C$=CHR$(
    31+119*F)                              :rem  85
220 IFS<256OR(S>40960ANDS<49152)ORS>53247
    THENGOSUB3000:GOTO210                  :rem 235
225 PRINT:PRINT:PRINT                      :rem 180
```

```
230 PRINT"[<5>]{ 2 SU} INDIRIZZO CONCLUSI
    VO{ 3 SPAZI}";:INPUTE:F=1-F:C$=CHR$(3
    1+119*F)                               :rem  92
240 IFE<256OR(E>40960ANDE<49152)ORE>53247
    THENGOSUB3000:GOTO230                  :rem 183
250 IFE<STHENPRINTC$;"{RVS}INDIRIZZO CONC
    LUSIVO<INDIRIZZO INIZIALE"             :rem 200
255 IFE<STHENGOSUB1000:GOTO230             :rem 119
260 PRINT:PRINT:PRINT                      :rem 179
300 PRINT"{CLR}";CHR$(14):AD=S:POKEV+21,0
                                           :rem 225
310 A=1:PRINTRIGHT$("0000"+MID$(STR$(AD),
    2),5);":";:FORJ=ATO6                   :rem 227
315 FORJ=ATO6                              :rem  33
320 GOSUB570:IFN=-1THENJ=J+N:GOTO320
                                           :rem 228
390 IFN=-211THEN710                        :rem  62
400 IFN=-204THEN790                        :rem  64
410 IFN=-206THENPRINT:INPUT"{GIU'} INSERI
    RE IL NUOVO INDIRIZZO";ZZ              :rem  13
414 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"
    {RVS}ESCE DAL CAMPO DI VALORI INDICAT
    O"                                     :rem 105
```

```
415 IFN=-206THENIFZZ<SORZZ>ETHENGOSUB1000
    :GOTO410                      :rem 202
417 IFN=-206THENAD=ZZ:PRINT:GOTO310
                                  :rem 238
420 IFN<>-196THEN480              :rem 133
430 PRINT:INPUT"LISTATO:DA";F:PRINT"
    { 9 SPAZI}A";:INPUTT          :rem 183
440 IFF<SORF>EORT<SORT>ETHENPRINT"MINIMO"
    ;S;" MASSIMO";E;"![<5>]":GOTO430
                                  :rem 147
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHT$("000
    0"+MID$(STR$(I),2),5);":";    :rem 30
451 FORK=0TO5:N=PEEK(I+K):PRINTRIGHT$("00
    "+MID$(STR$(N),2),3);",";     :rem 66
460 GETA$:IFA$>""THENPRINT:PRINT:GOTO310
                                  :rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRIN
    T:GOTO310                     :rem 50
480 IFN<0THENPRINT:GOTO310        :rem 168
490 A(J)=N:NEXTJ                  :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CK
    SUM=(CKSUM+A(I))AND255:NEXT   :rem 200
510 PRINTCHR$(18);:GOSUB570:PRINTCHR$(146
    );                            :rem 94
511 IFN=-1THENA=6:GOTO315         :rem 254
515 PRINTCHR$(20):IFN=CKSUMTHEN530
                                  :rem 122
520 PRINT:PRINT"{RED}LA LINEA E' STATA IN
    SERITA IN MANIERA"            :rem 157
525 PRINT"ERRATA. RIPETERE[<5>]":PRINT:GO
    SUB1000:GOTO310               :rem 27
530 GOSUB2000                     :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT:POKE54
    272,0:POKE54273,0             :rem 227
550 AD=AD+6:IFAD<ETHEN310         :rem 212
560 GOTO710                       :rem 108
570 N=0:Z=0                       :rem 88
580 PRINT"[<~>]";                 :rem 81
581 GETA$:IFA$=""THEN581          :rem 95
582 AV=-(A$="M")-2*(A$=",")-3*(A$=".")-4*
    (A$="J")-5*(A$="K")-6*(A$="L")
                                  :rem 41
583 AV=AV-7*(A$="U")-8*(A$="I")-9*(A$="O"
    ):IFA$="H"THENA$="0"          :rem 134
584 IFAV>0THENA$=CHR$(48+AV)      :rem 134
585 PRINTCHR$(20);:A=ASC(A$):IFA=13ORA=44
    ORA=32THEN670                 :rem 229
590 IFA>128THENN=-A:RETURN        :rem 137
600 IFA<>20THEN630                :rem 10
610 GOSUB690:IFI=1ANDT=44THENN=-1:PRINT"
    {SIN} {SIN}";:GOTO690         :rem 172
620 GOTO570                       :rem 109
630 IFA<48ORA>57THEN580           :rem 105
640 PRINTA$;:N=N*10+A-48          :rem 106
650 IFN>255THENA=20:GOSUB1000:GOTO600
                                  :rem 229
660 Z=Z+1:IFZ<3THEN580            :rem 71
670 IFZ=0THENGOSUB1000:GOTO570    :rem 114
680 PRINT",";:RETURN              :rem 240
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211)
                                  :rem 149
691 FORI=1TO3:T=PEEK(S%-I)        :rem 67
695 IFT<>44ANDT<>58THENPOKES%-I,32:NEXT
                                  :rem 205
700 PRINTLEFT$("{ 3 SIN}",I-1);:RETURN
                                  :rem 7
710 PRINT"{CLR}{RVS}*** SAVE ***
    { 3 GIU'}"                    :rem 236
715 PRINT"{ 2 GIU'}PREMERE {RVS}RETURN
    {OFF} PER USCIRE DAL SAVE{GIU'}"
                                  :rem 103
720 F$="":INPUT"{GIU'}NOME DEL FILE";F$:I
    FF$=""THENPRINT:PRINT:GOTO310
                                  :rem 42
730 PRINT:PRINT"{ 2 GIU'}{RVS}N{OFF}ASTRO
    O {RVS}D{OFF}ISCO: (N/D)"     :rem 128
740 GETA$:IFA$<>"N"ANDA$<>"D"THEN740
                                  :rem 30
750 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$:
    OPEN15,8,15,"S"+F$:CLOSE15    :rem 212
760 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256              :rem 3
762 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                  :rem 109
763 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                            :rem 69
765 K=S:POKE254,K/256:POKE253,K-PEEK(254)
    *256:POKE780,253              :rem 17
766 K=E+1:POKE782,K/256:POKE781,K-PEEK(78
    2)*256:SYS65496               :rem 235
770 IF(PEEK(783)AND1)OR(191ANDST)THEN780
                                  :rem 111
775 PRINT"{GIU'}FATTO.{GIU'}":GOTO310
                                  :rem 201
780 PRINT"{GIU'}ERRORE NEL SAVE-RIPROVA!"
    :IFDV=1THEN720                :rem 104
781 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO720          :rem 103
790 PRINT"{CLR}{RVS}*** LOAD ***
    { 2 GIU'}"                    :rem 212
795 PRINT"{ 2 GIU'}PREMERE {RVS}RETURN
    {OFF} PER USCIRE DAL LOAD{GIU'}"
                                  :rem 96
800 F$="":INPUT"{ 2 GIU'}NOME DEL FILE";F
    $:IFF$=""THENPRINT:GOTO310    :rem 115
810 PRINT:PRINT"{ 2 GIU'}{RVS}N{OFF}ASTRO
    O {RVS}D{OFF}ISCO: (N/D)"     :rem 127
820 GETA$:IFA$<>"N"ANDA$<>"D"THEN820
                                  :rem 28
830 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$
                                  :rem 157
840 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256              :rem 2
841 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                  :rem 107
845 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                            :rem 70
850 POKE780,0:SYS65493            :rem 11
860 IF(PEEK(783)AND1)OR(191ANDST)THEN870
                                  :rem 111
865 PRINT"{GIU'}FATTO.":GOTO310   :rem 184
870 PRINT"{GIU'}ERRORE NEL LOAD-RIPETI!
    {GIU'}":IFDV=1THEN800         :rem 19
880 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO800          :rem 102
1000 REM CICALINO                 :rem 231
1001 POKE54296,15:POKE54277,45:POKE54278,
     165                          :rem 207
1002 POKE54276,33:POKE54273,6:POKE54272,5
                                  :rem 42
1003 FORT=1TO200:NEXT:POKE54276,32:POKE54
     273,0:POKE54272,0:RETURN     :rem 202
2000 REM CAMPANELLO               :rem 130
2001 POKE54296,15:POKE54277,0:POKE54278,2
     47                           :rem 152
2002 POKE54276,17:POKE54273,40:POKE54272,
     0                            :rem 86
2003 FORT=1TO100:NEXT:POKE54276,16:RETURN
                                  :rem 57
3000 PRINTC$;"{RVS} NON IN PAGINA ZERO O
     SU{DES}ROM ":GOTO1000        :rem 240
```

REM:HW

Hardware

Il monitor del C16

di **A. Motta**

Fra i comandi del BASIC 3.5 in dotazione al Commodore C16 vi è l'istruzione "MONITOR", che consente di operare con l'Assembler del microprocessore 7501 (il cervello del C16), che, di fatto, è lo stesso Assembler del capofamiglia 6502.
Quando però l'utente, che dopo essersi impadronito del linguaggio BASIC decide di esplorare il mondo Assembler, sfoglia l'"USER MANUAL" per cercare le istruzioni da utilizzare con il comando MONITOR resta abbastanza deluso nel leggere quelle brevi righe che rimandano ad una fantomatica "COMMODORE PLUS/4 PROGRAMMER'S REFERENCE GUIDE", che alla data di stesura di questo articolo (inizio febbraio '85) risulta del tutto sconosciuta sul mercato.
In queste pagine cercherò di illustrare come usare il MONITOR di cui dispone il C16, premettendo che, per una maggior comprensione dell'articolo, è necessario avere almeno una conoscenza di base dell'Assembler del 6502. Per chi è completamente a digiuno della materia, ma desidera impadronirsene, segnalo che sono apparsi, su numeri precedenti della rivista, una serie di articoli di Alessandro Guida dal titolo "IMPARIAMO IL LINGUAGGIO MACCHINA CON IL VIC E C64" che trattano tale argomento.
Digitato il comando "MONITOR", e penso la maggior parte dei possessori del C16 abbia già provato ad eseguirlo, appaiono sul video i seguenti caratteri:

```
   PC  SR AC XR YR SP
; 0000 00 00 00 00 00
```

che, per chi non ha dimestichezza con il processore 6502, non hanno alcun significato, mentre rappresentano i registri interni di tale processore (per il C16 il 7501, come sopra accennato) e precisamente:
– PC = PROGRAM COUNTER: il registro che contiene la locazione di memoria con l'istruzione da eseguire;
– SR = STATUS REGISTER o registro dei FLAG
– AC = ACCUMULATOR: accumulatore o registro A
– XR = X REGISTER: registro indice X
– YR = Y REGISTER: registro indice Y
– SP = STACK POINTER: puntatore dello stack in pagina 1.
Prima di passare ad esaminare le possibilità operative offerte dal MONITOR conviene tener presente le seguenti regole:
- tutti i comandi vanno inseriti digitando la sola lettere iniziale (es. ASSEMBLER = A);
- i dati numerici susseguenti le istruzioni devono essere in notazione esadecimale;
- spaziare i comandi e i dati.
Ecco, di seguito, cosa potete fare una volta entrati nel modo MONITOR del vostro C16.

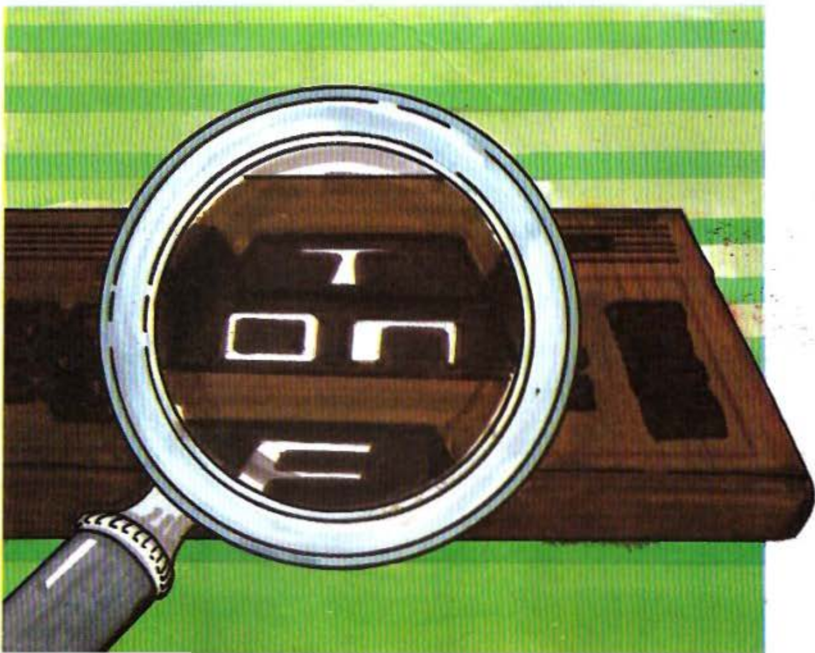
DISASSEMBLER
Disassembla un blocco di memoria, visualizzando i codici ASSEMBLY del 6502 contenuti nelle locazioni di memoria scelte.

Formato:
D <$xxxx (1)> <$xxxx(2)>
$xxxx(1) = indirizzo di partenza del blocco dati da disassemblare
$xxxx(2) = indirizzo fine blocco dati.

Esempio:
D 8000 83FF
appariranno sul video le istruzioni del linguaggio macchina contenute nel 1° K della ROM dell'interprete BASIC.

Se si esegue il comando D <$xxxx(1)> (seguito solamente dall'indirizzo di partenza), verranno disassemblate le prime 20 locazioni di memoria a partire da tale indirizzo.
Eseguendo solo il comando D vengono disassemblate le prime 20 locazioni di memoria a partire dall'ultimo indirizzo presente nel registro PC (Program Counter).
I dati, nella notazione esadecimale, vengono visualizzati nella seguente forma: a sinistra la locazione di memoria; poi il contenuto della stessa e, a destra, il codice mnemonico e l'operando della istruzione ASSEMBLY.

ASSEMBLER
Consente di assemblare un programma scritto in linguaggio macchina ed inserirlo in una determinata zona di memoria (RAM naturalmente).



Formato:
A <$xxxx(1)> <MNEMONICO> [<OPERANDO>]
$xxxx(1) = indirizzo dove sarà inserita l'istruzione
MNEMONICO = tipo di istruzione (es.: LDA – STA – TAX – .... ecc....)
OPERANDO = indirizzo o numero seguente l'istruzione

Esempio:
0300 LDA # $51
0302 STA $0C28
0305 BRK

quando mandato in esecuzione con il comando G (GOTO) più avanti illustrato, stamperà all'inizio della seconda riga del video il carattere cod. 81

Quando state utilizzando l'ASSEMBLY, il C16 provvede, ad inserimento della prima istruzione, a far comparire automaticamente a video la locazione d'inizio della successiva istruzione.

FILL
Provvede a riempire una zona di memoria con un valore desiderato.

Formato:
F <$xxxx(1)> <$xxxx(2)> <Byte>
$xxxx(1) = indirizzo di partenza del blocco da riempire
$xxxx(2) = indirizzo finale del blocco
Byte = valore codice da inserire nella zona di memoria sopra indicata.

Esempio:
F 0C28 0C4F 51 vi mostrerà a video, nella seconda linea, il carattere di cod. 81. Questo perché fra le locazioni 0C00 e 0FFF è ubicata la mappa video.

TRANSFER
Trasferisce il contenuto di una zona di memoria ad altra zona di memoria (per quest'ultima solo RAM).

Formato:
T <$xxxx(1)> <$xxxx(2)> <$xxxx(3)>
$xxxx(1) = locazione iniziale dove è contenuto il blocco dati da trasferire
$xxxx(2) = locazione finale del blocco da trasferire
$xxxx(3) = locazione iniziale della zona dove sarà trasferito il blocco dati

Esempio:
T FF81 FFFF 3000
trasferisce la KERNAL JUMP TABLE in RAM a partire dalla locazione 12288.

REGISTER
Visualizza lo stato dei registri interni del micro 7501 allo stesso modo di quando si entra nel "MONITOR", come in precedenza illustrato.

Formato: R

Per variare lo stato di tali registri basta portare il cursore sul contenuto degli stessi, inserire il valore desiderato e premere il tasto RETURN.

GOTO
Manda in esecuzione un programma in linguaggio macchina a partire dall'indirizzo indicato.

Formato:
G [<$xxxx>]
$xxxx =
indirizzo di partenza della routine in linguaggio macchina. Se omesso, la routine partirà dall'ultimo indirizzo contenuto nel registro PC (Program Counter). Quando viene incontrata l'istruzione BRK (Break) la routine si interrompe ed a video viene visualizzato lo stato dei registri del 7501, come indicato nel modo "R".

Esempio:
G 0300 (da eseguirsi dopo l'esempio del modo Assembler) manda in esecuzione la piccola routine in precedenza indicata.

Provate ora ad eseguire questo esempio: G 3FF6.
Piaciuto lo scherzetto? Non prendetevela, se avevate in memoria un programma che vi era costato fatica, ma avete imparato qualcosa di nuovo.

LOAD
Simile all'istruzione BASIC, carica un programma.
Formato:
L <"Filename"> <Device>
Filename = nome del programma da caricare scritto fra gli apici. Possibile ometterlo.
Device = unità da dove deve essere caricato il programma (1= Tape, 8 = Disco). Valore di default = 1

Con la sola istruzione L si caricherà il primo programma presente sul nastro. Naturalmente, una volta caricato il programma dovrete uscire dal MONITOR per utilizzarlo.

SAVE
Salva il contenuto di una zona di memoria sulla unità prescelta.

Formato:
S <"Filename"> <Device> <$xxxx(1)> <$xxxx(2)>
Filename = nome assegnato al blocco dati da registrare
Device = unità sulla quale registrare il blocco dati (1 = Tape, 8 = disco)
$xxxx(1) = indirizzo iniziale del blocco dati
$xxxx(2) = indirizzo finale del blocco dati.

A differenza dell'omonimo comando BASIC, il SAVE del MONITOR salva solamente il contenuto di una zona di memoria definita dall'utilizzatore con gli indirizzi $xxxx(1) e $xxxx(2), che obbligatoriamente devono essere indicati.
Pertanto, per registrare un programma scritto in BASIC con il SAVE del MONITOR bisognerà indicare l'indirizzo della locazione di memoria da dove ha inizio tale programma e l'indirizzo della locazione relativa alla fine dello stesso. Il primo, se non si è variato il contenuto delle locazioni $2B e $2C in pagina zero (che contengono il puntatore d'inizio area destinata al programma BASIC), è sempre $1001. Il secondo è invece quello indicato nelle locazioni $2D e $2E (sempre in pagina zero), che contengono la locazione di memoria indicante la fine del programma BASIC e l'inizio dell'area destinata alle variabili.
Perciò bisognerà prima leggere il contenuto di tali locazioni con il comando M 002B (più avanti illustrato), o con il comando D 002B 002E, e quindi dare al comando S i codici contenuti nelle locazioni $2B e $2C quale 1° indirizzo e quelli contenuti nelle locazioni $2D e $2E quale indirizzo finale del blocco.

Esempio:
S "PROVA", 1 1001 12FF

VERIFY
Verifica se la registrazione di un programma è avvenuta correttamente o vi sono degli errori.

Formato:
V <"Filename"> <Device>
Filename = Nome del programma (possibile ometterlo, nel qual caso verifica il primo programma che incontra sull'unità).
Device = Numero corrispondente all'unità collegata (1 = Tape, 8 = disco)
Analoga alla corrispondente istruzione BASIC.

COMPARE
Confronta il contenuto di due blocchi di memoria e visualizza le differenze.

Formato:
C <$xxxx(1)> <$xxxx(2)> <$xxxx(3)>
$xxxx(1) = Locazione di inizio del primo blocco da confrontare
$xxxx(2) = Locazione finale del primo blocco da confrontare
$xxxx(3) = Locazione d'inizio del secondo blocco.

HUNT
Ricerca una stringa o una sequenza di dati in una parte della memoria e visualizza la locazione iniziale dove sono ubicati i dati oggetto della ricerca.

Formato:
H <$xxxx(1)> <$xxxx(2)> <DATI>
$xxxx(1) = locazione iniziale della zona di memoria dove deve essere effettuata la ricerca
$xxxx(2) = locazione finale della suddetta zona di memoria
DATI = oggetto della ricerca. Se trattasi di una stringa, la stessa dovrà essere preceduta dal simbolo "'" apostrofo (Es. H 8000 FFFF 'READ e il MONITOR cercherà nella ROM la stringa

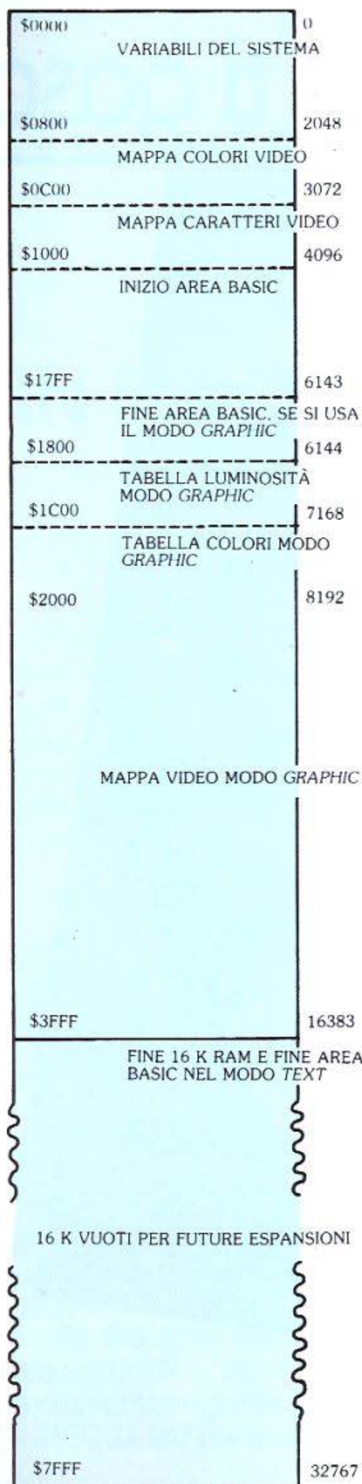


FIGURA 1/A MAPPA MEMORIA C16 DA 0 A 32K (RAM)

READ). Se trattasi invece di dati numerici, gli stessi dovranno essere spaziati l'uno dall'altro. (Es. H8000 FFFF A5 F4 C3).

(DUMP) MEMORY

Di fatto è il comando DUMP, ma visto che la D viene già usata per il DISASSEMBLER, viene chiamato MEMORY e si usa la M come comando. Visualizza il contenuto della zona di memoria indicata in righe da 8 byte per volta, che riportano (sulla destra dello schermo) i caratteri in campo inverso dei corrispondenti codici ASCII contenuti nelle singole locazioni oggetto del DUMP.
Va segnalato che i codici da 00 a 1F e da 80 a 9F, corrispondenti ai codici di controllo dei COMMODORE, vengono tutti visualizzati con un punto in campo inverso.

Formato:
M <$xxxx(1)> <$xxxx(2)>
$xxxx(1) = indirizzo iniziale del blocco oggetto del DUMP
$xxxx(2) = indirizzo finale di tale blocco

Esempio:
M 0000 00FF visualizzerà il DUMP della pagina zero.

Se viene eseguito solo il comando "M", verrà visualizzato il DUMP di 8 byte a partire dalla locazione di memoria contenuta nel registro PC.
Per modificare i valori dei byte oggetto del DUMP si può procedere in due modi. Il primo consiste nel portare il cursore lampeggiante sulla riga dove vi sono i byte da modificare; variare tali byte e premere il tasto RETURN. Il secondo nell'utilizzare il comando ">" (maggiore di) nel seguente:
Formato:
> <$xxxx> <BYTE (1)> [<BYTE (2).....(8)>]
$xxxx = locazione di memoria iniziale della variazione del DUMP
BYTE (1) = contenuto di tale locazione
BYTE (2)–(8) = possibilità di variare sino ad un massimo di 8 locazioni di memoria.

Esempio:
0300 FF FF FF FF FF mette nelle prime cinque locazioni della pagina 3 il valore FF (dec. 255)

Naturalmente, il comando deve essere utilizzato solo in presenza del MEMORY.

EXIT

Esce dal modo MONITOR e ritorna al BASIC.

Formato: X

Infine, per la gioia di tutti coloro che vogliono esplorare il proprio C16, viene riportata in figura 1) la mappa di memoria del computer.

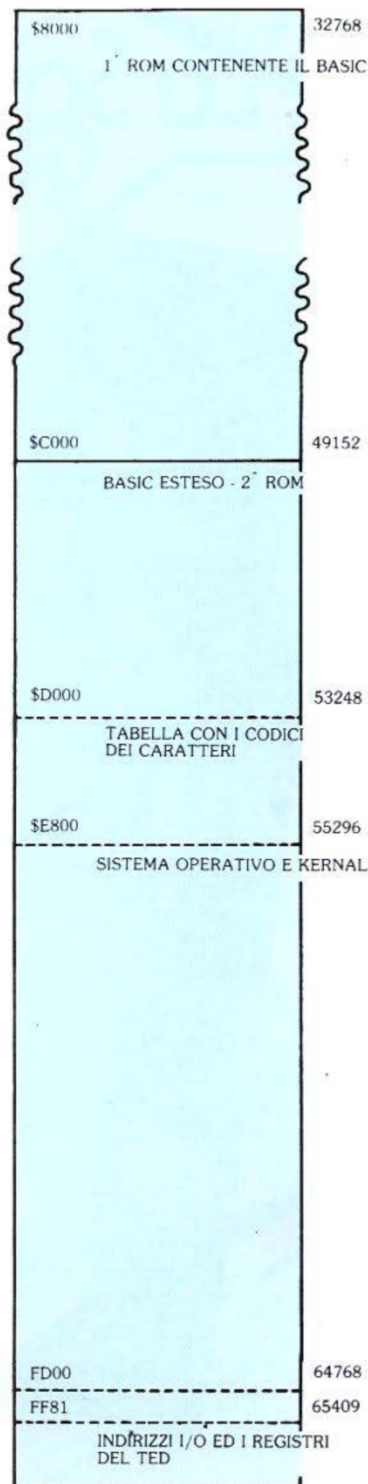
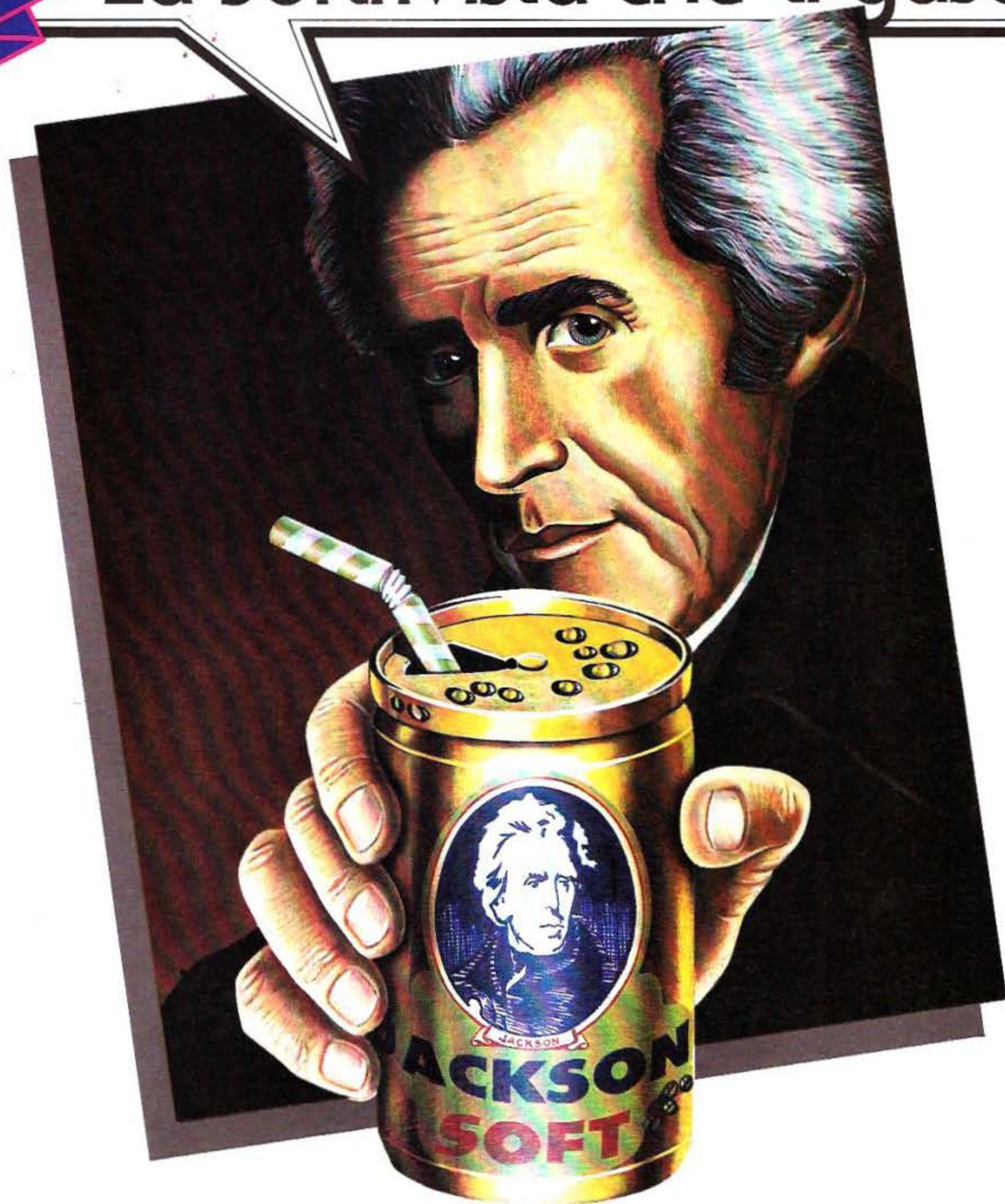


FIGURA 1/B MAPPA MEMORIA C16 DA 32 A 64K (BANCHI ROM)

FINALMENTE!

La Softrivista che ti gasa!



QUALCOSA DI SUPER, DI INEDITO, DI IRRESISTIBILE

IL VERO GIOCO COMINCIA ADESSO

È IN EDICOLA
JACKSON SOFT
SERIE ORO

La softrivista con i giochi esclusivi per Commodore 64 e Spectrum 48K importati dall'Inghilterra, mai presentati in Italia.
Una sfida Jackson al già visto, al già fatto, al... già registrato.



Ogni mese in edicola
- la softrivista con:
  6 pagine di commento al supergame presentato;
  4 pagine di rubriche;
  22 pagine di listati scelti tra il meglio di quanto pubblicato in fatto di grafica, giochi, utility.
- una cassetta originale con il SUPERGAME del mese

Tutto a sole 10.000 lire

QUO VADIS per C 64
Strategia, avventura, logica, inganno nel super computergame che sta facendo impazzire l'Inghilterra. Nessuno è ancora riuscito a raggiungere la fine del labirinto. Prova tu!

BRIAN BLOODAXE per Spectrum 48K
Un plat forme game che ti condurrà in pieno 21° secolo, alla conquista dei favolosi gioielli della corona inglese. Forza, astuzia, magia e... la colonna sonora di Monty Python.

Corri in edicola, il vero gioco comincia solo adesso e se sei davvero bravo partecipa alla "sfida al campione", utilizzando il tagliando che troverai sull'ultima pagina di copertina di ogni numero.



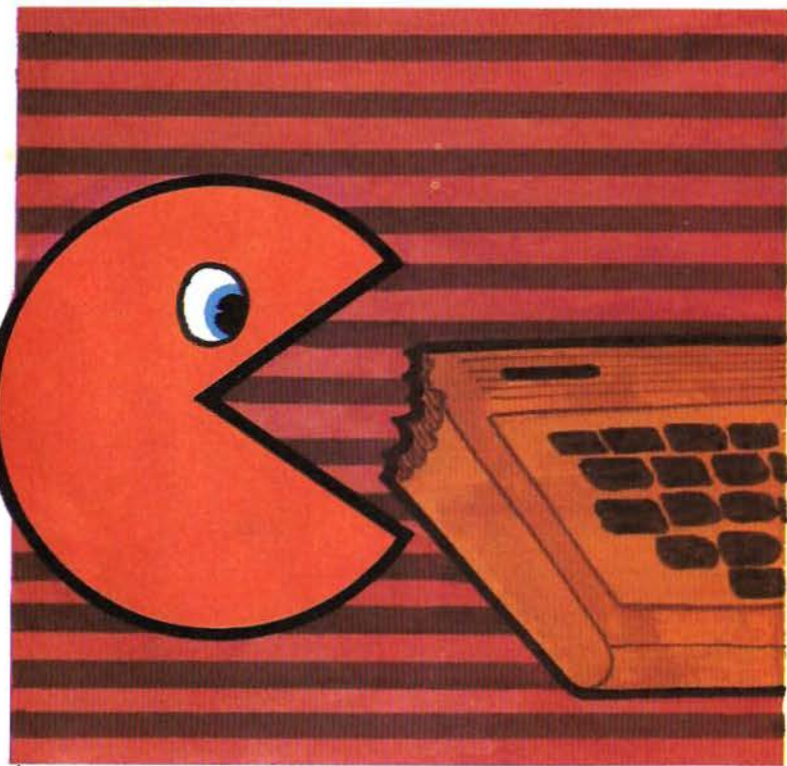
GRUPPO EDITORIALE JACKSON
SAN FRANCISCO • LONDRA • MILANO



Chomper

di **G. Hu**
trad. ed adatt.
di **M. Cristuib Grizzi**

In disperata ricerca di cristalli energetici la vostra astronave è atterrata su un pianeta alieno. I cristalli sono però sorvegliati da molti soldati e da un terribile droide. Muovetevi velocemente, perché il tempo scorre rapido... Un gioco eccitante per VIC 20 inespanso e per C64. È richiesto un joystick.



In questo gioco impersonate un esploratore spaziale proveniente dal pianeta Terra. Ad un certo punto del viaggio la vostra nave è incappata in un meteorite vagante e la quasi totalità della riserva di energia è stata impiegata nell'attivazione degli scudi protettivi. Vi trovate ora nei pressi di uno strano pianeta ricco di cristalli energetici, che possono ripristinare il livello di sicurezza nei serbatoi di energia della vostra nave.

Sfortunatamente, i cristalli sono sorvegliati da soldati e da pericolosi droidi, e quindi inviate sul pianeta un robot della serie Chomper per raccoglierli senza rischiare vite umane.

Usando il joystick dalla sala di controllo della nave, manovrate Chomper sulla superficie del pianeta per raccogliere il maggior numero di cristalli nel più breve tempo possibile. Per ogni cristallo raccolto ricevete 150 punti. Se incappate in un soldato, ne perdete 450 e se un droide raggiunge Chomper, quest'ultimo viene distrutto, provocando il fallimento della missione e la vostra morte.

Guerra contro il tempo

Disponete di 45 secondi per raccogliere il maggior numero di cristalli possibile. Se raggiungete i 3000 punti, verrete gratificati da tempo in bonus: il primo bonus è di 45 secondi, e ciascuno dei seguenti decrementa il successivo di 5 secondi. Se guadagnate 6 bonus, potrete disporre di un nuovo Chomper in un'area differente del pianeta, con 45 secondi di gioco aggiuntivi. Se desiderate più cristalli, premete il pulsante di fuoco, ma attenzione! Più cristalli significano anche più soldati!

Il gioco inizia con un diagramma delle posizioni del joystick: spingendo quest'ultimo verso destra si avrà una partita facile, mentre verso sinistra si avrà una partita difficile. Spingendo il joystick verso l'alto si incrementa il numero dei droidi: fino a sette nella versione per C64, e fino a due nella versione per VIC 20. Spingendo il joystick verso il basso si provoca la fine del gioco. Dopo la vostra selezione il campo di gioco ed i droidi vengono sistemati casualmente sullo schermo. Il punteggio, il tempo rimasto ed i bonus sono visualizzati nella parte alta dello schermo.

Alla fine del gioco, in funzione del punteggio raggiunto, vi sarà chiesto di inserire il vostro nome per la tabella dei massimi punteggi: digitate il vostro nome e premete RETURN. Apparirà un altro diagramma con il massimo punteggio nella parte alta dello schermo.

Note alla versione per VIC 20

Se trovate il gioco troppo difficile, oppure troppo facile, il livello di gioco può essere modificato in molte maniere. Il modo più semplice è modificare il tempo di bonus: cambiando il valore di R nella linea 390 si ottiene questo risultato. Modificando il 6 nell'ultima istruzione della linea 210 si cambia il numero di bonus richiesti per ottenere un nuovo schermo, mentre modificando il valore di F nelle linee 460-480 si può modificare il rapporto tra le mosse del Chomper e quelle dei droidi.

Note alla versione per C64

Con la differenza di permettere fino a sette droidi, la versione per C64 funziona esattamente come quella per VIC 20. Se avete un televisore in bianco e nero, premete il pulsante di fuoco all'inizio del gioco fino a che appaia una B. Ciò modifica i colori per renderli ben visibili anche su un apparecchio in bianco e nero o su un monitor monocromatico.

La versione per C64 è interamente in linguaggio macchina, e deve essere caricata da nastro posizionandosi con la cassetta in corrispondenza dell'inizio del programma e digitando:

LOAD"", 1,1

Il programma andrà quindi mandato in esecuzione con:

SYS 49152

Chi non disponesse della cassetta dovrà utilizzare MLX con i seguenti parametri:

indirizzo iniziale: 49152
indirizzo finale: 50891

Chomper versione per VIC 20

```
10 PRINT"{CLR}"SPC(183)"CHOMPER":FORI=1TO
   2000:NEXT:GOTO610                :rem 170
20 M=M+1:X=45+G-INT(TI/60):IFX<0THEN240
                                    :rem 44
30 PRINT"{RVS}{HOME}"SPC(7)"TEMPO"X"{SIN}
   ":GOSUB590:X=X+X1:Y=Y+Y1:IFY=0THENY=22
                                    :rem 146
40 IFY=23THENY=1                    :rem 228
50 IFX=-1THENX=21                   :rem 14
70 IF-((PAND32)=0)THEN510           :rem 127
80 POKEC,32:C=FNA(0):X1=X:Y1=Y:Z=PEEK(C):
   IFZ=32THENPOKEC,36:GOTO140       :rem 186
90 POKEC,37:POKEVO,8:IFZ=34THENS=S+150:GO
   SUB190:GOTO120                   :rem 215
100 IFZ=35THENS=S-450:GOSUB200:GOTO130
                                    :rem 74
110 POKEC,33:GOTO250                :rem 160
120 IFS>=LTHENPOKES2,0:POKES3,0:GOTO210
                                    :rem 204
130 POKES1,0:POKES2,0:POKES3,0:POKES4,0:P
    RINT"{RVS}{HOME}{DES}{SIN}"S"{SIN} "
                                    :rem 162
135 REM********ROUTINE DROIDI***********
                                    :rem 124
140 Z=M/F:IFZ<>INT(Z)THEN20         :rem 178
150 IFW=C1THENW=0                   :rem 32
160 W=W+1:X=X(W):Y=Y(W):X=X+(-(X1>X)+(X1<
    X)):Y=Y+(-(Y1>Y)+(Y1<Y)):POKED(W),32
                                    :rem 28
170 D(W)=FNA(0):X(W)=X:Y(W)=Y:Z=PEEK(D(W)
    ):POKED(W),33:IFZ>35THEN250     :rem 96
180 GOTO20                          :rem 52
185 REM*******ROUTINE SONORE************
                                    :rem 156
190 FORX=160TO198STEP3:POKES2,X:POKES3,X:
    NEXTX:RETURN                    :rem 49
200 POKES4,178:POKES1,178:FORX=150TO1STEP
    -3:POKEVO,(X/10):NEXT:POKEVO,8:RETURN
                                    :rem 1
205 REM**********BONUS*****************
                                    :rem 154
210 V=V+1:PRINT"{RVS}{HOME}"SPC(15)"*EX*
    { 2 SPAZI}{ 2 SIN}"V:L=S+2999:G=G+R:R
    =R-5:IFV<>6THEN20               :rem 231
220 PRINT"{CLR}{RVS}"SPC(5)"SCHERMO BONUS
    ":FORX=1TO750:NEXT:PRINT"{CLR}":M=0:G
    =0:R=45                         :rem 163
230 PRINT"{RVS}{HOME}"SPC(15)"*EX*
    { 2 SPAZI}{ 2 SIN}"V:GOTO510:rem 246
235 REM*********GAME OVER***************
                                    :rem 196
240 PRINT"{HOME}":FORX=1TO10:PRINT:NEXT:P
    RINT"{RVS}{ 5 SPAZI}FINE TEMPO":FORX=
    1TO1000:NEXT:GOTO260            :rem 56
250 FORX=230TO150STEP-1:POKES3,X:POKES1,X
    :NEXT:POKES3,0:POKES1,0         :rem 108
260 PRINT"{HOME}":FORX=1TO11:PRINT:NEXT:P
    RINTSPC(6)"{RVS}GAME OVER"      :rem 136
270 FORZ=1TO100STEP4:POKE36879,Z:POKES2,1
    28+ABS(INT(SIN(Z)*127)):NEXT:POKES2,0
                                    :rem 75
275 FORI=1TO1000:NEXT               :rem 25
280 POKE36879,110:PRINT"{CLR}":POKE36869,
    240                             :rem 113
285 REM********HIGH SCORE?**************
                                    :rem 250
290 IFS<BSTHEN310                   :rem 17
300 BS=S:PRINT"{CLR}"SPC(5)"GRAN PUNTEGGI
    O!":PRINT"{HOME}{ 3 GIU'} IL TUO NOME
    ?":INPUT" ";N$                  :rem 8
305 REM*********ANCORA?*****************
                                    :rem 179
310 FORI=1TO1000:NEXT               :rem 15
311 PRINT"{CLR}{ 3 GIU'}{ 5 DES}PUNTI:";S
    :PRINT"{ 3 GIU'}{ 2 DES}MIGLIORE :";B
    S:PRINTSPC(8)"DI"               :rem 246
320 PRINT"{ 2 DES}";N$:PRINT"{HOME}":FORZ
    =1TO11:PRINT:NEXT:PRINTSPC(6)"ANCORA?
    "                               :rem 4
330 PRINT:PRINTSPC(7)"SI{GIU'}"     :rem 73
340 PRINTSPC(8)"W":PRINTSPC(7)"W W NUOVO
    GIOCO":PRINTSPC(8)"W":PRINTSPC(8)"
    {GIU'}NO"                       :rem 86
350 GOSUB590:IFX=1THEN390           :rem 16
360 IFY=-1THENM=-1:GOTO390          :rem 66
370 IFY=1THENPOKE36879,27:PRINT"{CLR}
    {BLU}":END                      :rem 194
380 GOTO350                         :rem 108
385 REM*****INIZ.VARIABILI*************
                                    :rem 157
390 PRINT"{CLR}":POKE36869,240:POKE36879,
    110:X=0:Y=0:Z=0:L=3000:S=0:G=0:V=0:R=
    45:W=0                          :rem 14
400 X1=11:Y1=11:C=7933:DEF FNA(A)=7680+Y*
    22+X:DEF FNB(A)=INT(RND(TI)*A+1)
                                    :rem 73
410 S1=36874:S2=36875:S3=36876:S4=36877:V
    O=36878:D=37154                 :rem 110
415 PA=37137:PB=37152:POKE37139,0
                                    :rem 68
420 IFM<0THENM=0:GOTO510            :rem 208
425 REM********LIVELLO?***************
                                    :rem 197
430 M=0:PRINT"{CLR}{WHT}LIVELLO?
    { 3 SPAZI}":PRINT"{ 3 GIU'}
    { 3 SPAZI}DUE DROIDI":PRINT"{GIU'}"SP
    C(8)"W"                         :rem 199
440 PRINT"{ 2 SPAZI}FAC.' W W DIFF":PRINTS
    PC(8)"W":PRINT"{ 2 GIU'}{ 3 SPAZI}USA
    IL JOYSTICK"                    :rem 148
450 FORZ=1TO600:NEXT                :rem 250
460 GOSUB590:IFX=-1THENC1=1:F=3:GOTO500
                                    :rem 125
470 IFX=1THENC1=1:F=4:GOTO500       :rem 250
480 IFY=-1THENC1=2:F=2:GOTO500      :rem 40
```

```
490 GOTO460                                :rem 112
500 PRINT"{CLR}"                           :rem 249
505 REM******INIZ.SCHERMO***************
                                           :rem 105
510 POKE36869,255:FORA=34TO35:FORY=1TO70
                                           :rem 160
520 X=7701+FNB(484):Z=FNB(8)-1:IFZ=6THENZ
    =7                                     :rem 14
530 IFZ=2THENZ=3                           :rem 233
540 POKE30720+X,Z:POKEX,A:NEXTY,A
                                           :rem 58
550 POKEVO,8:FORX=128TO255:POKES2,X:NEXT:
    POKES2,0:POKEC,36:IFM<>0THEN20
                                           :rem 41
555 REM******INIZ.DROIDI****************
                                           :rem 66
560 FORZ=1TOC1:X=FNB(22)-1:Y=FNB(22):D(Z)
    =FNA(0):X(Z)=X:Y(Z)=Y                  :rem 214
570 POKED(Z),33:NEXT                       :rem 197
580 TI$="000000":GOTO20                    :rem 212
585 REM********JOYSTICK*****************
                                           :rem 16
590 POKED,127:P=PEEK(PA):X=-((PEEK(PB)AND
    128)=0)+((PAND16)=0):POKED,255
                                           :rem 95
600 Y=(-((PAND8)=0))+((PAND4)=0):RETURN
                                           :rem 191
605 REM********CARATTERI****************
                                           :rem 14
610 FORX=7424TO7431:POKEX,0:NEXT:FORY=33T
    O37:FORX=0TO7:READZ:POKE7168+8*Y+X,Z:
    NEXTX,Y                                :rem 10
630 DATA 60,66,90,66,60,36,36,102,0,0,0,2
    4,24                                   :rem 145
635 DATA 0,0,0,126,129,165,129,153,189,12
    9,126                                  :rem 221
640 DATA 60,126,90,126,235,129,213,126,60
    ,126,90,255,255,255,255,126:GOTO390
                                           :rem 46
```

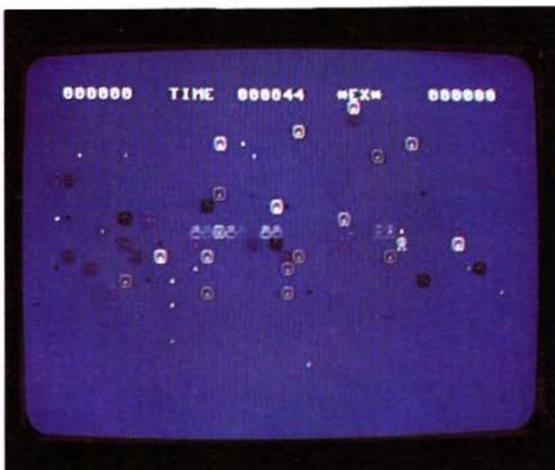


Chomper versione per C64

```
49152 :032,136,198,169,162,141,070
49158 :095,195,169,003,141,112,209
49164 :003,169,016,141,114,003,202
49170 :169,039,141,115,003,169,142
49176 :000,141,032,208,141,033,067
49182 :208,169,147,032,210,255,027
49188 :032,229,192,160,007,185,073
49194 :035,198,153,123,005,136,180
49200 :016,247,160,010,185,043,197
49206 :198,153,198,005,136,016,248
49212 :247,169,001,141,107,003,216
49218 :160,005,185,054,198,153,053
49224 :027,006,136,016,247,169,161
49230 :001,141,113,003,165,162,151
49236 :105,010,197,162,208,252,250
49242 :173,112,003,141,203,005,215
49248 :173,107,003,009,048,141,065
49254 :083,005,173,000,220,074,145
49260 :176,016,238,107,003,173,053
49266 :107,003,041,007,240,246,246
49272 :141,107,003,076,082,192,209
49278 :074,176,001,000,074,176,115
49284 :008,169,210,141,111,003,006
49290 :076,191,192,074,176,007,086
49296 :169,160,141,111,003,208,168
49302 :040,074,176,029,173,113,243
49308 :003,208,179,173,095,195,241
49314 :073,011,141,095,195,173,082
49320 :112,003,073,001,141,112,098
49326 :003,169,001,141,113,003,092
49332 :076,082,192,169,000,141,072
49338 :113,003,076,082,192,032,172
49344 :192,195,032,029,196,032,100
49350 :249,192,032,081,195,032,211
49356 :052,194,032,000,194,032,196
49362 :067,193,032,004,193,032,219
49368 :088,193,032,153,196,173,027
49374 :141,002,240,233,076,221,111
49380 :192,169,001,160,000,153,135
49386 :000,216,153,000,217,153,205
49392 :000,218,153,000,219,136,198
49398 :208,241,096,160,018,185,130
49404 :077,198,153,010,004,136,062
49410 :016,247,056,173,077,003,062
49416 :237,062,003,141,079,003,021
49422 :173,078,003,237,063,003,059
49428 :013,079,003,176,041,238,058
49434 :081,003,173,077,003,024,131
49440 :105,184,141,077,003,173,203
49446 :078,003,105,011,141,078,198
49452 :003,173,073,003,024,109,173
49458 :076,003,141,073,003,173,007
49464 :076,003,240,006,056,233,158
49470 :005,141,076,003,096,169,040
49476 :000,141,064,003,173,062,255
49482 :003,141,071,003,173,063,016
49488 :003,141,072,003,032,151,226
49494 :193,096,169,015,141,064,252
49500 :003,173,073,003,141,065,038
49506 :003,169,000,141,066,003,224
49512 :032,168,193,173,081,003,242
49518 :141,065,003,169,000,141,117
49524 :066,003,169,032,141,064,079
49530 :003,032,168,193,096,206,052
49536 :075,003,208,018,206,073,199
49542 :003,173,073,003,201,255,074
49548 :208,003,076,130,197,169,155
49554 :009,141,075,003,096,173,131
```

```
49560 :071,003,056,233,016,141,160
49566 :065,003,173,072,003,233,195
49572 :039,141,066,003,169,000,070
49578 :141,067,003,141,068,003,081
49584 :141,069,003,162,015,014,068
49590 :065,003,046,066,003,120,229
49596 :248,173,067,003,109,067,087
49602 :003,141,067,003,173,068,137
49608 :003,109,068,003,141,068,080
49614 :003,173,069,003,109,069,120
49620 :003,141,069,003,216,088,220
49626 :202,016,216,162,002,189,237
49632 :067,003,072,074,074,074,076
49638 :074,032,244,193,104,041,150
49644 :015,032,244,193,202,016,170
49650 :236,096,009,048,238,064,165
49656 :003,172,064,003,153,000,131
49662 :004,096,165,162,105,005,023
49668 :197,162,208,252,032,127,214
49674 :193,096,165,162,197,162,217
49680 :240,252,096,072,138,024,070
49686 :101,251,133,251,165,252,151
49692 :105,000,133,252,104,096,206
49698 :072,134,254,165,251,056,198
49704 :229,254,133,251,165,252,044
49710 :233,000,133,252,104,096,096
49716 :160,000,173,060,003,133,069
49722 :251,173,061,003,133,252,163
49728 :169,032,145,251,173,000,066
49734 :220,074,176,005,162,040,235
49740 :032,034,194,074,176,005,079
49746 :162,040,032,019,194,074,091
49752 :176,005,162,001,032,034,242
49758 :194,074,176,005,162,001,194
49764 :032,019,194,074,176,003,086
49770 :032,081,195,165,251,233,039
49776 :039,141,079,003,165,252,023
49782 :233,004,013,079,003,176,114
49788 :013,165,251,024,105,192,106
49794 :133,251,165,252,105,003,015
49800 :133,252,165,251,233,232,122
49806 :141,079,003,165,252,233,247
49812 :007,013,079,003,144,013,151
49818 :165,251,056,233,192,133,160
49824 :251,165,252,233,003,133,173
49830 :252,177,251,201,035,208,010
49836 :042,173,062,003,024,105,069
49842 :150,141,062,003,173,063,002
49848 :003,105,000,141,063,003,243
49854 :201,253,144,003,076,182,025
49860 :198,169,038,141,070,003,047
49866 :169,129,141,004,212,169,002
49872 :009,141,001,212,076,063,198
49878 :195,162,128,142,004,212,033
49884 :206,109,003,174,109,003,056
49890 :142,001,212,224,010,208,255
49896 :005,162,034,142,011,212,030
49902 :162,037,142,070,003,201,085
49908 :036,208,065,056,173,062,076
49914 :003,233,209,141,079,003,150
49920 :173,063,003,233,040,013,013
49926 :079,003,176,013,169,016,206
49932 :141,062,003,169,039,141,055
49938 :063,003,076,040,195,173,056
49944 :062,003,056,233,194,141,201
49950 :062,003,173,063,003,233,055
49956 :001,141,063,003,169,035,192
49962 :141,011,212,169,020,141,224
49968 :008,212,141,109,003,076,085
49974 :063,195,201,034,208,003,246
49980 :076,130,197,160,000,173,028
49986 :070,003,145,251,165,251,183
49992 :141,060,003,165,252,141,066
49998 :061,003,096,160,000,173,059
50004 :027,212,074,074,074,074,107
50010 :074,201,006,240,244,169,000
50016 :001,153,040,216,153,000,147
50022 :217,153,000,218,153,230,049
50028 :218,200,208,227,162,027,126
50034 :032,127,193,202,208,250,102
50040 :162,070,169,232,133,253,115
50046 :169,003,133,254,160,004,081
50052 :173,027,212,024,101,253,154
50058 :133,253,165,254,105,000,024
50064 :133,254,136,208,239,165,255
50070 :253,056,233,040,165,254,127
50076 :233,004,144,218,160,000,147
50082 :173,027,212,016,004,169,251
50088 :035,208,002,169,036,145,251
50094 :253,032,012,194,202,208,051
50100 :197,160,039,169,001,153,131
50106 :000,216,136,016,250,096,132
50112 :120,165,001,041,251,133,135
50118 :001,160,000,185,000,208,240
50124 :153,000,056,185,000,209,039
50130 :153,000,057,136,208,241,237
50136 :165,001,009,004,133,001,017
50142 :173,024,208,041,240,009,149
50148 :014,141,024,208,088,160,095
50154 :039,185,245,195,153,016,043
50160 :057,136,016,247,096,060,084
50166 :066,090,066,060,036,036,088
50172 :102,000,000,000,024,024,146
50178 :000,000,000,126,129,165,166
50184 :129,153,189,129,126,060,026
50190 :126,090,126,235,129,213,165
50196 :126,060,126,090,255,255,164
50202 :255,255,126,169,147,032,242
50208 :210,255,169,006,141,032,077
50214 :208,141,033,208,169,016,045
50220 :141,062,003,169,039,141,087
50226 :063,003,169,204,141,060,178
50232 :003,169,005,141,061,003,182
50238 :169,048,141,073,003,169,153
50244 :045,141,076,003,169,009,255
50250 :141,075,003,169,199,141,034
50256 :077,003,169,050,141,078,086
50262 :003,169,000,141,081,003,227
50268 :162,008,169,006,157,093,175
50274 :003,173,027,212,157,083,241
```

```
50280 :003,202,016,242,169,003,227
50286 :141,108,003,160,023,169,202
50292 :000,153,000,212,136,016,121
50298 :250,169,015,141,024,212,165
50304 :169,017,141,005,212,141,045
50310 :012,212,169,245,141,006,151
50316 :212,141,013,212,169,129,248
50322 :141,015,212,141,018,212,117
50328 :096,169,000,141,082,003,131
50334 :173,027,212,205,111,003,121
50340 :176,003,076,069,197,173,090
50346 :060,003,172,061,003,032,245
50352 :084,197,142,104,003,141,079
50358 :103,003,172,082,003,185,218
50364 :083,003,133,251,185,093,168
50370 :003,168,133,252,165,251,142
50376 :032,084,197,142,106,003,252
50382 :141,105,003,169,032,160,048
50388 :000,145,251,173,105,003,121
50394 :205,103,003,240,015,144,160
50400 :008,162,001,032,034,194,143
50406 :076,238,196,162,001,032,167
50412 :019,194,173,106,003,205,168
50418 :104,003,240,015,144,008,244
50424 :162,040,032,034,194,076,018
50430 :005,197,162,040,032,019,197
50436 :194,160,000,177,251,201,219
50442 :037,144,003,076,130,197,085
50448 :201,034,208,013,172,082,214
50454 :003,185,083,003,133,251,168
50460 :185,093,003,133,252,172,098
50466 :082,003,165,251,153,083,003
50472 :003,165,252,153,093,003,197
50478 :160,000,169,034,145,251,037
50484 :165,251,024,105,000,133,218
50490 :251,165,252,105,212,133,152
50496 :252,169,001,145,251,238,096
50502 :082,003,173,082,003,205,106
50508 :107,003,240,003,076,184,177
50514 :196,096,141,079,003,152,237
50520 :056,233,004,141,080,003,093
50526 :162,000,173,080,003,208,208
50532 :007,173,079,003,201,040,091
50538 :144,021,173,079,003,056,070
50544 :233,040,141,079,003,173,013
50550 :080,003,233,000,141,080,143
50556 :003,232,076,096,197,096,056
50562 :032,229,192,169,032,160,176
50568 :200,153,039,004,136,208,108
50574 :250,056,173,062,003,237,155
50580 :114,003,141,079,003,173,149
50586 :063,003,237,115,003,013,076
50592 :079,003,144,012,173,062,121
50598 :003,141,114,003,173,063,151
50604 :003,141,115,003,169,001,092
50610 :141,113,003,169,000,141,233
50616 :032,208,141,033,208,141,179
50622 :008,212,160,008,185,011,006
50628 :198,153,055,004,136,016,246
50634 :247,160,014,185,020,198,002
50640 :153,132,004,136,016,247,128
50646 :160,009,185,060,198,153,211
50652 :211,004,136,016,247,173,239
50658 :114,003,141,071,003,173,219
50664 :115,003,141,072,003,169,223
50670 :220,141,064,003,032,151,081
50676 :193,173,000,220,041,016,119
50682 :208,008,173,113,003,208,195
50688 :244,076,023,192,169,000,192
50694 :141,113,003,240,234,007,232
50700 :001,013,005,032,015,022,100
50706 :005,018,032,016,018,005,112
50712 :019,019,032,020,018,009,141
50718 :007,007,005,018,032,030,129
50724 :032,004,018,015,009,004,118
50730 :019,005,001,019,025,060,171
50736 :032,062,008,001,018,004,173
50742 :022,032,017,021,009,020,175
50748 :032,008,009,032,019,003,163
50754 :015,018,005,032,003,008,147
50760 :015,013,016,005,018,020,159
50766 :009,013,005,032,032,032,201
50772 :032,032,032,032,032,032,020
50778 :032,032,042,005,024,042,011
50784 :025,015,021,032,001,018,208
50790 :005,032,015,014,005,032,205
50796 :015,006,032,020,008,005,194
50802 :032,006,005,023,032,020,232
50808 :015,032,023,009,014,032,245
50814 :001,020,032,003,008,015,205
50820 :013,016,005,018,032,029,245
50826 :196,169,000,160,150,153,198
50832 :063,003,136,208,250,032,068
50838 :229,192,160,006,185,070,224
50844 :198,153,200,005,136,016,096
50850 :247,160,014,185,020,198,218
50856 :153,020,006,136,016,247,234
50862 :173,000,220,041,016,208,064
50868 :249,096,169,147,032,210,059
50874 :255,032,229,192,160,039,069
50880 :185,096,198,153,184,005,245
50886 :136,016,247,076,130,162,197
```





Kablam!

di **S. Ressler**
trad. ed adatt.
di **M. Cristuib Grizzi**

Un pazzo arroccato sul tetto del vostro palazzo sta gettanto bombe incendiarie sulla strada sottostante. Il vostro compito consiste nel manovrare una tinozza piena d'acqua per raccogliere e disattivare le bombe incendiarie. Un gioco "arcade style" in versione per VIC 20 inespanso e per C64. E' richiesto un joystick.



Il vostro solitamente quieto quartiere viene improvvisamente disturbato da una serie di esplosioni e di fiammate. Fortunatamente intervenite voi, muniti di una tinozza, e intercettate prontamente tutte le bombe che un pazzo lancia sulla strada.

Quando appare la schermata iniziale vi verrà chiesto di scegliere tra 15 diversi livelli di difficoltà, dove 1 è il più facile e 15 il più difficile.

Usando un joystick (Porta 2 per il C64) spostate la tinozza a destra e sinistra per raccogliere le bombe prima che tocchino il suolo. La tinozza può uscire dal bordo dello schermo e rientrare dal bordo opposto: ciò è estremamente utile giocando ai livelli più difficili.

Se una bomba raggiunge il suolo, esploderà e voi perderete una delle vostre quattro tinozze. Il gioco ha termine soltanto quando avete terminato le tinozze a disposizione; a questo punto potrete scegliere un altro livello di difficoltà e giocare ancora.

Se l'azione diventa troppo frenetica, o se desiderate una pausa nel bel mezzo del gioco, premete la barra spaziatrice. Il gioco rimarrà "congelato" fino alla pressione del tasto CTRL.

Note alla versione per VIC 20

La versione per VIC 20 è simile a quella per C64, ma contiene alcune differenze. Avete tre tinozze invece di quattro e cinque livelli di difficoltà invece di quindici. In ogni caso, la velocità di caduta delle bombe aumenta per ogni bomba che riuscite ad intercettare, e diminuisce per ogni bomba che vi sfugge e raggiunge il suolo.

La versione per VIC 20 visualizza inoltre il massimo punteggio raggiunto, il punteggio corrente, la velocità delle bombe e le tinozze rimaste. Il tasto SHIFT LOCK è usato come tasto di pausa. Il programma, pur essendo totalmente in linguaggio macchina, si carica da cassetta normalmente (senza bisogno dell'indirizzo secondario 1) e si attiva con un semplice RUN.

Chi non disponesse della cassetta allegata alla rivista dovrà inserire un'espansione di memoria da almeno 8 Kbyte,

accendere il computer e digitare in modo diretto la seguente linea:

POKE642,32:SYS58232

Quindi occorrerà caricare MLX e utilizzare come parametri:
indirizzo iniziale: 6291
indirizzo finale: 7682

Una volta registrato il programma su nastro o disco non sarà più necessaria l'espansione di memoria per il suo caricamento.

Kablam! versione per VIC 20

```
6291 :011,016,000,000,158,052,128
6297 :049,048,057,000,000,000,051
6303 :169,000,141,092,003,141,193
6309 :093,003,169,024,141,015,098
6315 :144,032,130,020,032,100,117
6321 :016,032,012,018,032,074,105
6327 :017,169,007,141,074,003,082
6333 :169,003,141,064,003,173,230
6339 :076,003,016,024,032,235,069
6345 :017,172,074,003,196,251,146
6351 :208,003,076,077,016,032,107
6357 :055,019,032,030,019,169,025
6363 :010,141,076,003,206,076,219
6369 :003,032,140,017,032,131,068
6375 :016,032,078,019,032,219,115
6381 :019,173,072,003,208,207,151
6387 :076,245,018,169,000,141,124
6393 :060,003,141,061,003,141,146
6399 :074,003,141,108,021,141,231
6405 :064,003,032,074,017,169,108
6411 :003,141,072,003,169,010,153
6417 :141,076,003,096,169,239,229
6423 :141,019,145,173,017,145,151
6429 :041,016,208,003,206,108,099
6435 :021,169,127,141,034,145,160
6441 :173,032,145,041,128,208,000
6447 :003,238,108,021,173,108,186
6453 :021,041,015,141,108,021,144
6459 :208,006,201,002,176,002,142
6465 :169,001,201,011,144,002,081
6471 :169,012,141,064,003,168,116
6477 :162,002,185,162,031,208,059
6483 :008,169,032,153,162,031,126
6489 :032,055,017,185,184,031,081
6495 :208,008,169,032,153,184,081
6501 :031,032,055,017,185,206,115
6507 :031,208,008,169,032,153,196
6513 :206,031,032,055,017,200,142
6519 :202,016,213,172,064,003,021
6525 :196,252,240,071,164,252,020
6531 :169,032,162,003,153,184,066
6537 :031,153,206,031,200,202,192
6543 :016,246,172,064,003,169,045
6549 :027,153,184,031,153,206,135
6555 :031,132,252,169,000,153,124
6561 :184,151,153,206,151,200,182
6567 :169,028,153,184,031,153,117
6573 :206,031,169,000,153,184,148
6579 :151,153,206,151,200,169,185
6585 :029,153,184,031,153,206,173
6591 :031,169,000,153,184,151,111
6597 :153,206,151,096,173,082,034
6603 :003,240,003,206,082,003,228
6609 :238,060,003,173,060,003,234
6615 :208,003,238,061,003,162,122
6621 :004,160,016,024,032,240,185
6627 :255,174,060,003,173,061,185
6633 :003,032,205,221,056,173,155
6639 :060,003,237,092,003,133,255
6645 :170,173,061,003,237,093,214
6651 :003,005,170,144,012,173,246
6657 :060,003,141,092,003,173,217
6663 :061,003,141,093,003,162,214
6669 :008,160,016,024,032,240,237
6675 :255,174,092,003,173,093,041
6681 :003,032,205,221,096,169,239
6687 :206,133,167,169,031,133,102
6693 :168,173,141,002,208,251,212
6699 :169,228,133,170,169,031,175
6705 :133,171,169,228,133,180,039
6711 :169,151,133,181,162,021,104
6717 :160,021,177,167,208,012,038
6723 :072,169,032,145,167,104,244
6729 :145,170,169,000,145,180,114
6735 :136,016,237,056,165,167,088
6741 :233,022,133,167,165,168,205
6747 :233,000,133,168,056,165,078
6753 :170,233,022,133,170,165,222
6759 :171,233,000,133,171,056,099
6765 :165,180,233,022,133,180,254
6771 :165,181,233,000,133,181,240
6777 :202,016,193,096,032,148,040
6783 :224,165,142,162,003,010,065
6789 :202,208,252,024,101,142,038
6795 :010,101,142,105,003,133,121
6801 :142,074,074,056,233,012,224
6807 :176,252,105,015,133,251,059
6813 :096,169,128,141,019,145,087
6819 :169,255,141,034,145,169,052
6825 :147,032,210,255,162,011,218
6831 :160,005,024,032,240,255,123
6837 :160,000,185,244,019,032,053
6843 :210,255,200,192,012,208,240
6849 :245,169,014,032,210,255,094
6855 :162,013,160,005,024,032,083
6861 :240,255,160,000,185,000,021
6867 :020,032,210,255,200,192,096
6873 :012,208,245,032,228,255,173
6879 :240,251,201,049,144,247,075
6885 :201,054,176,243,056,233,168
6891 :048,073,255,141,063,003,050
6897 :169,147,032,210,255,032,062
6903 :202,020,174,063,003,024,221
6909 :169,000,105,048,202,208,217
```

```
6915 :251,141,082,003,162,003,133
6921 :160,016,024,032,240,255,224
6927 :160,000,185,062,020,032,218
6933 :210,255,200,192,007,208,069
6939 :245,162,006,160,016,024,128
6945 :032,240,255,160,000,185,137
6951 :069,020,032,210,255,200,057
6957 :192,012,208,245,162,010,106
6963 :160,016,024,032,240,255,010
6969 :160,000,185,081,020,032,023
6975 :210,255,200,192,007,208,111
6981 :245,162,013,160,016,024,177
6987 :032,240,255,160,000,185,179
6993 :088,020,032,210,255,200,118
6999 :192,015,208,245,162,022,163
7005 :160,001,024,032,240,255,037
7011 :160,000,185,111,020,032,095
7017 :210,255,200,192,019,208,165
7023 :245,169,000,141,019,145,062
7029 :173,017,145,041,032,208,221
7035 :249,160,021,169,032,153,139
7041 :228,031,136,208,248,096,052
7047 :160,002,162,022,024,032,025
7053 :240,255,160,000,185,012,225
7059 :020,032,210,255,200,192,032
7065 :018,208,245,032,228,255,115
7071 :240,251,201,089,240,007,163
7077 :201,078,240,006,076,010,008
7083 :019,076,029,016,000,164,219
7089 :251,140,074,003,162,001,040
7095 :024,032,240,255,160,000,126
7101 :185,030,020,032,210,255,153
7107 :200,192,017,208,245,096,129
7113 :172,074,003,162,001,024,125
7119 :032,240,255,160,000,185,055
7125 :047,020,032,210,255,200,209
7131 :192,014,208,245,096,160,110
7137 :021,185,228,031,208,028,158
7143 :169,036,153,228,031,169,249
7149 :000,153,228,151,072,152,225
7155 :072,032,182,019,032,219,031
7161 :019,104,168,104,206,072,154
7167 :003,076,116,019,136,208,045
7173 :220,169,032,153,228,031,070
7179 :162,011,160,016,024,032,160
7185 :240,255,173,082,003,205,207
7191 :084,003,240,013,160,000,011
7197 :185,103,020,032,210,255,066
7203 :200,192,008,208,245,173,037
7209 :082,003,141,084,003,056,154
7215 :073,255,170,169,000,032,234
7221 :205,221,162,015,160,016,064
7227 :024,032,240,255,174,072,088
7233 :003,169,000,032,205,221,183
7239 :096,169,015,141,014,144,138
7245 :169,129,141,013,144,162,067
7251 :010,138,072,032,219,019,061
7257 :104,170,202,208,246,169,164
7263 :000,141,014,144,024,173,079
7269 :082,003,105,016,141,082,018
7275 :003,096,172,082,003,162,113
7281 :080,202,208,253,136,192,160
7287 :255,208,246,056,173,013,046
7293 :144,233,012,009,128,141,024
7299 :013,144,096,144,032,032,080
7305 :075,065,066,076,065,077,049
7311 :032,032,032,076,069,086,214
7317 :069,076,032,040,049,045,204
7323 :053,041,063,080,076,065,021
7329 :089,032,065,071,065,073,044
7335 :078,032,089,032,079,082,047
7341 :032,078,063,031,032,157,054
7347 :017,094,095,157,157,017,204
7353 :033,035,017,157,157,144,216
7359 :017,064,144,032,157,017,110
7365 :032,032,157,157,017,032,112
7371 :032,017,157,157,017,028,099
7377 :083,067,079,082,069,144,221
7383 :030,072,073,017,157,157,209
7389 :083,067,079,082,069,144,233
7395 :156,083,080,069,069,068,240
7401 :144,066,079,077,066,083,236
7407 :017,157,157,157,157,157,017
7413 :076,069,070,084,032,032,096
7419 :032,032,157,157,157,157,175
7425 :080,082,069,083,083,032,174
7431 :070,073,082,069,032,084,161
7437 :079,032,083,084,065,082,182
7443 :084,160,000,185,000,128,064
7449 :153,000,028,200,208,247,093
7455 :160,000,185,000,129,153,146
7461 :000,029,200,208,247,160,113
7467 :007,185,020,021,153,000,173
7473 :028,136,016,247,160,039,163
7479 :185,028,021,153,216,028,174
7485 :136,016,247,160,007,185,044
7491 :068,021,153,008,029,136,226
7497 :016,247,160,031,185,076,020
7503 :021,153,024,029,136,016,202
7509 :247,169,255,141,005,144,022
7515 :096,162,021,169,059,133,219
7521 :167,169,030,133,168,160,156
7527 :006,169,037,145,167,024,139
7533 :165,168,105,120,133,168,200
7539 :169,002,145,167,056,165,051
7545 :168,233,120,133,168,136,055
7551 :208,231,024,165,167,105,003
7557 :022,133,167,165,168,105,125
7563 :000,133,168,202,208,213,039
7569 :160,015,169,038,153,088,000
7575 :030,136,016,248,160,015,244
7581 :169,002,153,088,150,136,087
7587 :016,248,096,048,008,008,075
7593 :060,126,126,126,060,255,154
7599 :255,255,063,031,015,007,033
7605 :003,255,255,255,255,255,179
```

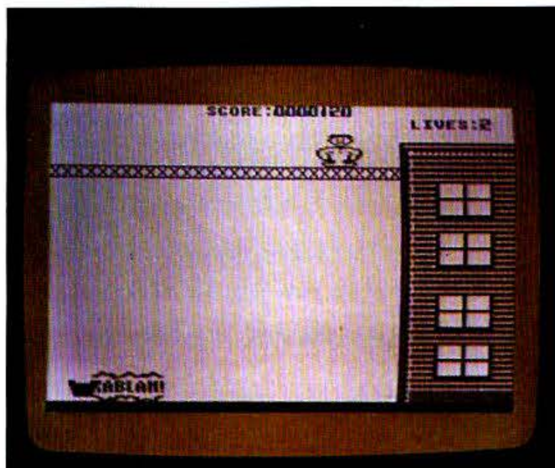
```
7611 :255,255,255,255,255,255,181
7617 :252,248,240,224,192,015,084
7623 :018,016,010,005,060,064,116
7629 :072,240,072,008,080,160,069
7635 :056,004,036,072,072,072,011
7641 :072,119,064,064,063,036,123
7647 :036,036,036,220,004,004,047
7653 :248,200,084,156,212,051,156
7659 :043,048,035,238,238,238,051
7665 :000,187,187,187,000,255,033
7671 :065,036,024,024,036,065,241
7677 :255,058,013,013,013,210,047
```

Kablam! versione per C64

```
100 FORX=24573TO24593:CS=CS+PEEK(X):NEXT:
    IFCS=2907THEN160                   :rem   5
110 PRINT"{CLR}{ 5 GIU'}STO LEGGENDO:"
                                       :rem 192
120 PRINT"LA ROUTINE L/M";:FORX=24573TO25
    219:READA:POKEX,A:B=B+A:NEXT:rem     32
130 IFB<>72932THENPRINT" ERRORE NEI DATI!
    ":END                              :rem 229
140 PRINT" OK":PRINT"CARATTERI":FORX=1024
    0TO10575:READA:POKEX,A:NEXT :rem     58
150 PRINT"SPRITE":FORX=12800TO13056:READA
    :POKEX,A:NEXT                      :rem 207
160 POKE53281,1                        :rem  38
170 GOTO460:REM VA ALLA PAGINA TITOLO
                                       :rem 197
180 POKE53272,27:PRINT"{CLR}[<5>]{GIU'}";
    CHR$(8):POKE53280,7                :rem  58
190 PRINTTAB(30)"{RED}{GIU'}'$$$$$$$$"
                                       :rem 142
200 PRINTTAB(30)"{SU}#KKKKKKKK":rem      69
210 FORX=0TO1:PRINTTAB(30)"{SU}#KKKKKKKK
    "                                  :rem   0
220 PRINTTAB(30)"{SU}#KKMRVRNKK":rem    101
230 PRINTTAB(30)"{SU}#KKX&%&WKK":rem    241
240 PRINTTAB(30)"{SU}#KKPTUTOKK":rem    110
250 PRINTTAB(30)"{SU}#KKKKKKKKK":rem     74
260 PRINTTAB(30)"{SU}#KKMRVRNKK":rem    105
270 PRINTTAB(30)"{SU}#KKX&%&WKK":rem    245
280 PRINTTAB(30)"{SU}#KKPTUTOKK":rem    114
290 PRINTTAB(30)"{SU}#KKKKKKKKK":NEXT
                                       :rem 199
300 PRINTTAB(30)"{SU}#[<8>]JJJJJJJJ
    {HOME}"                            :rem 235
310 PRINT"{HOME}{ 5 GIU'}[<7>]LLLLLLLLLL
    LLLLLLLLLLLLLLLLL"                 :rem  79
320 PRINT"{HOME}"TAB(13)"YZ[£]!@@@@@@@
    {GIU'}"SPC(5)"(-)]Y!C"             :rem 124
330 POKEV+21,31                        :rem  57
340 POKEV,128:POKEV+1,226:POKEV+2,20:POKE
    V+3,69:POKEV+4,30:POKEV+5,69:rem    141
350 POKE830,1:POKE832,255:POKEV+30,0
                                       :rem  54
360 SYS24573                           :rem 157
370 A1=210:A2=30:A3=40:A4=215:A5=190:POKE
    V+21,26                            :rem 100
380 POKE24838,2:FORQ=1TO10             :rem  21
390 POKEV+8,INT(RND(1)*A1)+A2:POKEV+9,INT
    (RND(1)*A1)+A3                     :rem 184
400 POKEV+6,INT(RND(1)*A4)+A2:POKEV+7,INT
    (RND(1)*A5)+A3                     :rem 179
410 SYS24897:NEXT                      :rem  27
420 FORX=VTOV+9:POKEX,0:NEXT:POKEV+21,0
                                       :rem   4
430 FORI=1TO2000:NEXT:PRINT"{CLR}":POKE53
    272,21                             :rem 172
440 PRINT"{HOME}{ 12 GIU'}"TAB(15)"GAME O
    VER":FORI=1TO2000:NEXT:GOTO160
                                       :rem  48
450 REM PAGINA TITOLO*******************
                                       :rem  80
460 V=53248:S=2040:SI=54272:PRINT"{CLR}
    {BLU}":POKE53280,7                 :rem 101
470 POKEV+39,9:POKEV+40,12:POKEV+41,0:POK
    EV+29,26:POKEV+43,8:POKEV+42,15
                                       :rem  36
480 POKES,200:POKES+1,202:POKES+2,201:POK
    ES+4,203:POKES+3,203               :rem  67
490 POKEV+21,12:POKEV+23,8:POKESI+24,15
                                       :rem  26
500 POKEV+4,160:POKEV+5,50:POKESI+6,240
                                       :rem  20
510 FORK=50TO130:POKEV+5,K:POKESI+1,255-K
    :POKESI,100:POKESI+4,17:NEXT:rem     11
520 POKESI+1,0:POKESI,0:POKESI+4,0
                                       :rem  59
530 POKEV+21,8:POKEV+6,153:POKEV+7,122
                                       :rem 214
540 POKE24838,255:SYS24897:OPEN1,0
                                       :rem 217
550 PRINT"{CLR}{ 14 GIU'}"             :rem 236
560 PRINT"{GIU'}"TAB(8)"[<8>]LIVELLO DIFF
    ICOLTA'(1-15){ 3 SPAZI}{ 3 SIN}";
                                       :rem 246
570 INPUT#1,SL$:PRINT:SL=VAL(SL$):IFSL>15
    ORSL<1THEN550                      :rem 197
580 POKE53281,1:CLOSE1:SK=ABS(SL-16)*15+1
    5:POKE24838,SK                     :rem 217
590 POKEV+6,0:POKEV+7,0:POKEV+23,0:POKEV+
    42,8                               :rem 188
600 GOTO180                            :rem 104
610 REM****DATI LINGUAGGIO MACCHINA******
                                       :rem  91
620 DATA 162,0,169,10,157,192,7 :rem    244
630 DATA 169,0,157,192,219,232,224
                                       :rem 143
640 DATA 40,208,241,120,169,98,141
                                       :rem 139
650 DATA 21,3,169,66,141,20,3    :rem   138
660 DATA 88,32,149,97,76,168,96 :rem     28
670 DATA 169,0,170,168,24,109,167
                                       :rem 100
680 DATA 2,144,1,200,202,208,247:rem     25
690 DATA 152,73,255,141,0,208,173
                                       :rem  90
```

```
700 DATA 30,208,41,5,201,5,240    :rem  174
710 DATA 3,76,97,96,172,3,208     :rem  160
720 DATA 140,5,208,174,2,208,232:rem   30
730 DATA 232,232,232,232,232,232,232
                                  :rem  221
740 DATA 232,232,142,4,208,32,162
                                  :rem   78
750 DATA 97,169,255,141,64,3,76 :rem   10
760 DATA 97,96,165,197,201,60,240
                                  :rem  106
770 DATA 3,76,49,234,169,0,141   :rem  204
780 DATA 24,212,173,1,220,201,251
                                  :rem   71
790 DATA 240,2,208,247,169,15,141
                                  :rem   93
800 DATA 24,212,76,49,234,165,251
                                  :rem   92
810 DATA 240,18,174,2,208,169,1 :rem  243
820 DATA 133,251,232,224,225,240,6
                                  :rem  127
830 DATA 142,2,208,76,190,96,174:rem   50
840 DATA 2,208,169,0,133,251,202:rem   29
850 DATA 224,17,240,226,142,2,208
                                  :rem   82
860 DATA 76,190,96,172,3,208,140:rem   50
870 DATA 5,208,174,2,208,232,232:rem   38
880 DATA 232,232,232,232,232,232,232
                                  :rem  227
890 DATA 232,142,4,208,172,5,208:rem   40
900 DATA 200,192,223,240,12,140,5
                                  :rem   67
910 DATA 208,32,255,96,32,39,97 :rem    7
920 DATA 76,128,96,120,173,77,4 :rem    4
930 DATA 240,35,206,77,4,174,4  :rem  198
940 DATA 208,142,8,208,162,0,142:rem   34
950 DATA 4,208,172,5,208,140,9  :rem  197
960 DATA 208,32,65,97,169,255,141
                                  :rem  109
970 DATA 9,208,141,64,3,88,76   :rem  165
980 DATA 168,96,234,76,43,98,173:rem   74
990 DATA 62,3,141,60,3,169,235  :rem  200
1000 DATA 141,61,3,206,61,3,208 :rem  223
1010 DATA 251,206,60,3,208,241,169
                                  :rem  126
1020 DATA 5,141,62,3,169,64,141 :rem  235
1030 DATA 96,3,206,96,3,173,96  :rem  204
1040 DATA 3,208,245,96,162,32,142
                                  :rem   83
1050 DATA 6,212,174,64,3,202,142:rem   23
1060 DATA 64,3,142,1,212,162,10 :rem  221
1070 DATA 142,0,212,162,17,142,4:rem   18
1080 DATA 212,96,32,149,97,169,240
                                  :rem  152
1090 DATA 141,13,212,162,4,169,8:rem   32
1100 DATA 141,8,212,169,100,141,7
                                  :rem   69
1110 DATA 212,169,129,141,11,212,160
                                  :rem  218
1120 DATA 15,140,62,3,32,255,96 :rem  236
1130 DATA 169,4,141,8,212,169,129
                                  :rem   90
1140 DATA 141,11,212,202,208,220,169
                                  :rem  215
1150 DATA 5,141,8,212,169,129,141
                                  :rem   83
1160 DATA 11,212,160,160,140,62,3
                                  :rem   63
1170 DATA 32,255,96,162,200,142,13
                                  :rem  129
1180 DATA 212,169,2,141,62,3,32 :rem  233
1190 DATA 255,96,202,208,242,32,149
                                  :rem  193
1200 DATA 97,96,162,0,232,169,0 :rem  244
1210 DATA 157,0,212,224,22,208,246
                                  :rem  122
1220 DATA 96,173,24,4,201,9,240 :rem  237
1230 DATA 4,238,24,4,96,173,23  :rem  194
1240 DATA 4,201,9,240,24,169,0  :rem  181
1250 DATA 141,24,4,238,23,4,206 :rem  232
1260 DATA 6,97,206,6,97,206,6   :rem  157
1270 DATA 97,206,6,97,206,6,97  :rem  216
1280 DATA 96,173,22,4,201,9,240 :rem  241
1290 DATA 12,169,0,141,23,4,141 :rem  230
1300 DATA 24,4,238,22,4,96,173  :rem  191
1310 DATA 21,4,201,9,240,12,169 :rem  227
1320 DATA 0,141,22,4,141,23,4   :rem  117
1330 DATA 238,21,4,96,173,20,4  :rem  189
1340 DATA 201,9,240,15,169,0,141:rem   24
1350 DATA 21,4,8,49,4,141,23    :rem   86
1360 DATA 4,238,20,4,96,173,19  :rem  199
1370 DATA 4,201,9,240,15,169,0  :rem  185
1380 DATA 141,20,4,141,21,4,141 :rem  221
1390 DATA 22,4,238,19,4,96,162  :rem  202
1400 DATA 0,169,0,157,19,4,232  :rem  185
1410 DATA 224,7,208,248,96,120,169
                                  :rem  145
1420 DATA 234,141,21,3,169,49,141
                                  :rem   81
1430 DATA 20,3,88,173,21,208,41 :rem  236
1440 DATA 18,141,21,208,76,149,97
                                  :rem   98
1450 DATA 72,152,72,138,72,173,0:rem   38
1460 DATA 220,41,4,208,11,173,167
                                  :rem   79
1470 DATA 2,173,167,2,105,4,141 :rem  235
1480 DATA 167,2,173,0,220,41,8  :rem  187
1490 DATA 208,9,56,173,167,2,233:rem   49
1500 DATA 4,141,167,2,104,170,104
                                  :rem   68
1510 DATA 168,104,76,35,96,0,0  :rem  193
1520 DATA 0,2,0,0,0,46,0        :rem  118
1530 DATA 0,0,0,0,128,0,119     :rem   17
1540 DATA 20,255,227            :rem  213
1550 REM*******DATI CARATTERI***********
                                  :rem  142
1560 DATA 126,102,102,230,230,230,254,0,2
     4,24,24,56,56,56,56,0,254  :rem  152
1570 DATA 198,6,254,224,224,254,0,254,6,6
     ,62,14,14,254,0,198,198    :rem   93
1580 DATA 198,254,14,14,14,0,254,192,192,
     254,14,206,254,0,254,198   :rem  139
1590 DATA 192,254,230,230,254,0,254,198,1
     2,24,56,56,56,0,126,102    :rem   76
1600 DATA 102,254,230,230,254,0,254,198,1
     98,254,14,206,254,0,255    :rem   74
1610 DATA 255,255,255,255,255,255,255,239
     ,239,239,0,127,127,127,0   :rem  150
1620 DATA 255,66,36,24,24,36,66,255,255,2
     55,192,192,192,192,192     :rem   61
1630 DATA 192,255,255,3,3,3,3,3,3,3,3,3,3
     ,3,3,255,255,192,192,192   :rem  108
1640 DATA 192,192,192,255,255,192,192,192
     ,192,192,192,192,192       :rem  225
1650 DATA 255,255,0,0,0,0,0,0,3,3,3,3,3,3
     ,3,3,0,0,0,0,0,0,255       :rem   90
1660 DATA 255,24,24,24,24,24,24,255,255,2
     55,255,24,24,24,24,24      :rem  233
1670 DATA 24,3,3,3,255,255,3,3,3,192,192,
     192,255,255,192,192        :rem  144
1680 DATA 192,0,60,102,48,12,102,60,0,0,6
     0,102,96,96,102,60,0       :rem  150
1690 DATA 0,60,102,102,102,102,60,0,0,124
     ,102,102,124,102,102,0     :rem  208
```

```
1700 DATA 0,126,96,120,96,96,126,0,0,102,
     102,126,102,102,102,0,0     :rem  28
1710 DATA 126,24,24,24,24,126,0,0,0,0,0,0
     ,0,0,0,0,24,24,0,0,24       :rem 143
1720 DATA 24,0,102,102,102,102,0,0,0,0,22
     0,222,220,216,220,222       :rem 155
1730 DATA 220,216,255,255,0,255,255,119,3
     4,0,24,24,24,255,255        :rem 182
1740 DATA 24,24,24,0,0,0,255,255,0,0,0,25
     5,255,192,223,222,220       :rem 197
1750 DATA 216,216,0,96,96,96,96,96,124,0,
     0,102,102,102,102,60,24,0  :rem 159
1760 *******DATI SPRITE*****************
                                 :rem 225
1770 DATA 0,0,0,0,0,0,3,128,7,3,192,15,3,
     255,255,3,255,255,1         :rem  98
1780 DATA 255,254,1,255,254,0,255,252,0,2
     55,252,0,255,252,0,127      :rem  22
1790 DATA 248,0,127,248,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0,0,0,0,0        :rem 165
1800 DATA 0,0,0,0,0,0,0,0,0,0,0,0,128,0
     ,4,32,0,1,128,0,3,8,0,6     :rem 212
1810 DATA 32,0,6,0,0,15,0,0,15,0,0,63,192
     ,0,255,240,1,255,248,3      :rem 228
1820 DATA 255,252,3,255,252,3,255,252,3,2
     55,252,1,255,248,0          :rem  85
1830 DATA 255,240,0,127,224,0,31,128,0,0,
     127,0,0,255,128,0,65        :rem 149
1840 DATA 0,0,148,128,0,128,128,0,93,0,0,
     34,0,0,28,0,1,255           :rem 251
1850 DATA 128,6,8,96,8,0,16,8,201,16,9,0,
     144,8,129,48,4,129          :rem  95
1860 DATA 32,4,129,32,3,195,192,1,231,128
     ,2,195,64,2,0,64,1          :rem  75
1870 DATA 255,128,0,16,8,8,58,28,92,111,5
     4,246,195,99,227,1          :rem 115
1880 DATA 128,0,0,0,164,200,149,164,168
     ,149,170,169,93,202         :rem 137
1890 DATA 201,93,206,169,213,170,169,85,1
     70,169,84,170,205,85        :rem 218
1900 DATA 0,0,0,0,0,0,192,236,3,98,187,6,
     55,147,116,29,1,220         :rem 111
1910 DATA 8,0,136,0,0            :rem 247
```



LIBRI FIRMATI JACKSON

Rita Bonelli
Luciano Pazzucconi
Fabio Racchi
Giovanni Valerio

Commodore 64
la grafica e il suono

Ogni argomento viene spiegato e accompagnato da numerosi esempi commentati. Nel libro sono listati moltissimi programmi che sono registrati sulla cassetta allegata.
Nel Capitolo 1 si tratta del video, della tastiera e della grafica in modo carattere. Il Capitolo 2 è dedicato alla grafica e ne approfondisce tutti gli aspetti. Il Capitolo 3 tratta degli sprite.. Nel Capitolo 4 viene trattato l'argomento del suono. Completano il libro tre appendici: i registri del VIC II e del SID e le frequenze delle note musicali.

Cod. 409 B Pag. 270 Lire 34.000

Rita Bonelli
Commodore 64
i file

Questo libro tratta in maniera completa e precisa la gestione dei file su cassetta e su disco.
Oltre a brevi programmi esempio, riportati per spiegare l'uso delle istruzioni, il libro contiene cinque programmi per creare e gestire un archivio di dati: SEQUENZIALE su cassetta, SEQUENZIALE su disco, RANDOM su disco, RANDOM/USER su disco, RELATIVO su disco.

Cod. 400 B Pag. 173 Lire 17.000

Rita Bonelli
Commodore 64
il basic

Un'accurata esposizione del linguaggio BASIC, accompagnata da numerosi esempi.
Nel Capitolo 1 si ha una panoramica dei diversi argomenti. Il Capitolo 2 è dedicato al linguaggio. Nel Capitolo 3 si approfondisce l'uso della tastiera e del video. Il Capitolo 4 fornisce le informazioni necessarie per usare disco e cassetta per memorizzare programmi.
Il Capitolo 5 è dedicato alla stampante. Nel Capitolo 6 si parla della costruzione del programma.
Nel Capitolo 7 vengono passati in rassegna i codici e i numeri del calcolatore. Il Capitolo 8 è dedicato alla memoria. Nel Capitolo 9 si tratta degli errori. Completano il libro, l'Appendice A dedicata alla tastiera e l'Appendice B all'argomento del BASIC compilato.
Tutti i programmi esempio riportati nel libro sono disponibili a richiesta su floppy disk.

Cod. 348 D Pag. 316 Lire 26.000

COMMODORE 64 la grafica e il suono — Rita Bonelli, Luciano Pazzucconi
COMMODORE 64 i file — Rita Bonelli
COMMODORE 64 il basic

CON CASSETTA

GRUPPO EDITORIALE JACKSON

LA BIBLIOTECA CHE FA TESTO.

ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON – Divisione Libri - Via Rosellini, 12 - 20124 Milano

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
		Totale	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 3000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ____________
N° ____________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura
Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|

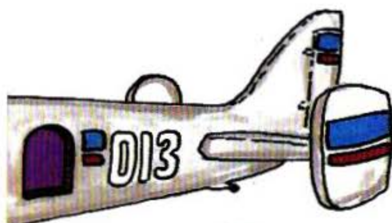
ORDINE MINIMO L. 50.000



Paratrooper

di **J. Goetz**
trad. ed adatt.
di **M. Cristuib Grizzi**

Un gioco di alta responsabilità: controllate il destino di dieci paracadutisti, dando il segnale di lancio che li fa gettare dall'aereo in volo. Il loro buon atterraggio dipende dalla vostra abilità nel valutare i fattori peso, vento, posizione e quota dell'aereo.





Quasi tutti abbiamo assistito nella nostra vita ad un'esibizione di paracadutisti, e sappiamo quanto questo sport richieda tempismo e precisione nei calcoli. Nel gioco ci sono tre piattaforme di atterraggio di differenti dimensioni: la più piccola è quella che rende di più in termini di punteggio, ma è più difficile da raggiungere. Mentre nel mare intorno alle piattaforme veleggiano spensierate barchette, voi a bordo dell'aereo dovete calcolare attentamente la forza del vento (visualizzato come FV in alto sullo schermo), il peso del paracadutista (visualizzato come PS), la quota a cui si trova l'aereo e la sua velocità, prima di dare il segnale di lancio premendo un tasto qualunque. Una volta lanciato il paracadutista non sarà più controllabile in alcun modo.
Le tre piattaforme, dalla più grande alla più piccola, se raggiunte dal paracadutista fruttano rispettivamente 25, 50 e 75 punti. Ovviamente, per ciascun lancio variano le condizioni di vento, peso, quota e velocità dell'aereo. Potete inoltre selezionare il livello di gioco desiderato tra due diverse opzioni (tre nella versione per C64).

Paratrooper versione per VIC 20

programma 1

```
10 POKE52,27:POKE56,27:CLR:I=6912
                                   :rem  162
15 PRINT"{CLR}{ 3 GIU'}{ 5 DES}ATTENDERE.
   .."                             :rem  180
20 READA:IFA=256THEN35             :rem   58
30 CH=CH+A:POKEI,A:I=I+1:GOTO20 :rem  123
35 IFCH<>21476THENPRINT"ERRORE NEI DATI":
   END                             :rem  222
40 S$=CHR$(131)                    :rem  255
50 FORI=1TOLEN(S$):POKE630+I,ASC(MID$(S$,
   I)):NEXT:POKE198,I:END          :rem   92
6000 I=6912:IFPEEK(I)=120THENRETURN
                                   :rem  133
6020 READA:IFA=256THENRETURN       :rem   24
6030 POKEI,A:I=I+1:GOTO6020        :rem   78
6912 DATA 120,169,13,141,20,3,169,27,141,
     21,3,88                       :rem   55
6918 DATA 96,169,1,240,11,206,14,27,169,1
     10,141,15                     :rem  162
6924 DATA 144,76,21,235,173,4,144,208,251
     ,169,32,141                   :rem   11
6930 DATA 37,145,169,130,141,36,145,238,1
     4,27,169,59                   :rem   26
```

```
6936 DATA 141,15,144,198,0,208,9,160,44,3
     2,171,27                       :rem  118
6942 DATA 165,251,133,0,198,1,208,9,160,8
     8,32,171                       :rem  121
6948 DATA 27,165,252,133,1,198,2,208,9,16
     0,110,32                       :rem  116
6954 DATA 171,27,165,253,133,2,206,232,3,
     208,73,173                     :rem  216
6960 DATA 233,3,141,232,3,172,234,3,169,3
     2,153,0                        :rem   50
6966 DATA 30,200,153,0,30,206,234,3,16,36
     ,169,20                        :rem   48
6972 DATA 141,234,3,173,20,145,77,24,145,
     74,74,74                       :rem  126
6978 DATA 74,74,74,168,185,194,27,141,107
     ,27,141,111                    :rem   33
6984 DATA 27,141,160,27,141,166,27,76,168
     ,27,172,234                    :rem   28
6990 DATA 3,169,2,153,0,30,200,169,3,153,
     0,30                           :rem  152
6996 DATA 76,191,234,162,21,185,73,31,133
     ,254,185,72                    :rem   30
7002 DATA 31,153,73,31,136,202,208,246,16
     5,254,153,73                   :rem   48
7008 DATA 31,96,0,22,44,66,88,256
                                    :rem  101
```

programma 2

```
0 PRINT"{CLR}{ 6 GIU'}{ 5 DES}{BLK}{RVS}
  PARATROOPER"                      :rem  124
1 PRINT"{ 3 GIU'}{RED}{ 7 SPAZI}{RVS}(N)O
  VIZIO":PRINT"{ 2 GIU'}{RED}{ 7 SPAZI}
  {RVS}(E)SPERTO"                   :rem  176
2 B1$="{RED}7{GIU'}{SIN}5{SU}":B2$="{PUR}
  50{GIU'}{ 2 SIN}//{SU}":B3$="{WHT}/2/
  {GIU'}{ 3 SIN}/5/{SU}":E$="/{GIU'}{SIN}
  /{SU}":Q$="{HOME}{ 20 GIU'}"  :rem  236
3 GETA$:IFA$="N"THENB1$=B1$+E$:B2$=B2$+E$
  :B3$=B3$+E$:GOTO5                 :rem  195
4 IFA$<>"E"THEN3                    :rem  141
5 PRINT"{CLR}":FORA=38400TO38905:POKEA,0:
  NEXT:FORA=38752TO38773:POKEA,5:NEXT:C=3
  0720                              :rem  243
6 PRINT"{CLR}":FORA=38400TO38905:POKEA,0:
  NEXT:FORA=38752TO38773:POKEA,5:NEXT:C=3
  0720                              :rem  244
7 FORA=38796TO38817:POKEA,2:NEXT:FORA=388
  40TO38861:POKEA,7:NEXT            :rem  181
8 FORA=1TO9:READSO(A):NEXT:DATA 175,195,2
  07,215,215,207,215,215,0          :rem   95
10 POKE36869,255:FORI=7168TO7223:READA:PO
   KEI,A:NEXT                       :rem  133
15 FORA=7552TO7632:POKEA,PEEK(A+26624):NE
   XT:FORA=7544TO7551:POKEA,255:NEXT
                                    :rem   27
16 FORA=7424TO7431:POKEA,0:NEXT :rem  142
20 DATA 60,126,126,255,255,255,129,90,90,
   60,24,24,32,36,66,0              :rem  134
25 DATA 14,17,127,255,1,0,0,0,3,7,255,255
   ,248,248,120,56                  :rem  181
27 DATA 0,1,3,7,8,63,31,15,128,192,224,24
   0,176,248                        :rem  150
28 DATA 240,224,195,36,24,219,60,24,24,24
                                    :rem  224
30 POKE1002,20:POKE1001,10:SYS6912
                                    :rem  166
31 PRINT"{HOME}{ 13 GIU'}{ 2 DES}"B1$"
   { 6 DES}"B2$"{ 5 DES}"B3$    :rem  238
35 PRINT"{HOME}{ 16 GIU'}{GRN} DE{BLK}":P
   RINT"{GIU'}{RED} DE":TR=10:SC=0:SQ=799
   6                                :rem    9
40 WT=INT(RND(1)*125+75):WS=INT(RND(1)*9+
   1):POKE198,0:POKESQ,32:POKESQ+C,0
                                    :rem  165
42 POKESQ+22,32:POKESQ+C+22,0:FORA=38730T
   O38751:POKEA,6:NEXT              :rem  147
45 POKE251,20-WS:POKE252,18-WS:POKE0,20-W
   S:POKE1,18-WS                    :rem    7
47 PRINT"{HOME}{ 4 GIU'}{BLK}{ 22 SPAZI}
   "                                :rem   35
50 PRINTQ$"{BLK}{RVS}PT{ 2 DES}="SC"{SIN}
   { 2 SPAZI}":PRINTQ$"{ 10 DES}{RVS}UOMI
   NI="TR"{SIN} "                   :rem   40
51 PRINTQ$"{GIU'}{RVS}PS{ 2 SPAZI}="WT"
   {SIN} ":PRINTQ$"{RVS}{GIU'}{ 10 DES}VE
   NTO ="WS"{SIN} "                 :rem   84
52 IFTR=0THEN300                    :rem  203
55 GETA$:IFA$=""THEN55              :rem  247
60 SX=PEEK(1002):SY=PEEK(7019)/22+1:DX=WS
   /20:DY=WT/400                    :rem  176
70 POKESQ,32:POKESQ+22,32:SP=SX+7680+INT(
   SY)*22                           :rem   94
72 CL=PEEK(SP+30742)AND15:CO=PEEK(SP+3072
   0)AND15:IFCL<>0ORCO<>0THEN90 :rem  171
75 POKESP,0:POKESP+22,1             :rem  146
80 SX=SX+DX:SY=SY+DY:SQ=SP:FORA=1TO100:NE
   XT                               :rem  148
85 GOTO70                           :rem   13
90 IFCL=2ANDSY<13THENSC=SC+75:GOSUB200:GO
   TO40                             :rem  103
100 IFCL=1ANDSY<13THENSC=SC+25:GOSUB200:G
    OTO40                           :rem  137
110 IFCL=4ANDSY<13THENSC=SC+50:GOSUB200:G
    OTO40                           :rem  139
190 PRINT"{HOME}{ 4 GIU'}{BLK}{ 2 SPAZI}
    {RVS}{ 4 SPAZI}PECCATO!!{ 5 SPAZI}":T
    R=TR-1:POKESQ,6:POKESQ+C,1:FORV=15TO0
    STEP-1                          :rem  148
195 POKE36877,210:POKE36878,V:FORTD=1TO50
    :NEXT:NEXT:POKE36877,0:SC=SC-10:GOTO4
    0                               :rem  148
200 POKESQ,0:POKESQ+22,1:PRINT"{HOME}
    { 4 GIU'}{BLK} {RVS}MISSIONE COMPIUTA
    { 3 SPAZI}"                     :rem  135
210 POKE36878,15:FORA=1TO9:POKE36876,SO(A
    ):FORB=1TO130:NEXT:NEXT:RETURN
                                    :rem  251
300 PRINT"{HOME}{ 4 GIU'}{BLK}{ 7 SPAZI}
    {RVS}GAME OVER":PRINT"{GIU'}
    { 5 SPAZI}{RVS}ANCORA?"         :rem  212
310 POKE37166,127:POKE788,191:POKE789,234
    :POKE37166,192                  :rem  117
320 GETA$:IFA$="S"THENRUN           :rem    0
330 IFA$<>"N"THEN320                :rem   90
```



Paratrooper versione per C64

```
100 PRINT"{CLR}{ 5 GIU'}"TAB(13)"ATTENDER
    E PREGO...":JS=56320            :rem 209
110 FORI=1TO1016:READA:CH=CH+A:NEXT I
                                    :rem 237
120 IFCH<>67163THENPRINT"ERRORE NEI DATI!
    ":END                           :rem  45
130 RESTORE                         :rem 184
140 GOSUB930                        :rem 177
150 A$="{HOME}{ 39 SPAZI}"          :rem 143
160 SC=0:TR=10:MB=53246:XP=53248:YP=XP+1:
    XA=YP+1:YA=XA+1:JS=56320:SD=12288
                                    :rem 173
170 PR=2040:EN=53269:CD=53278:CL=194:SH=1
    95:PL=193:PA=192:CR=53287:S=54272
                                    :rem 227
180 GOSUB630:GOSUB370:FORI=53250TO53256ST
    EP2:POKEI,INT(RND(0)*255):NEXT
                                    :rem 128
190 FORI=YPTOYP+14STEP2:READA:POKEI,A:NEX
    T                               :rem  41
200 GOSUB690                        :rem 177
210 GOSUB650:GOSUB530               :rem   0
220 POKEEN,254:POKE53276,224:POKE53258,35
    :POKE53260,170:POKEMB,32:POKE53262,50
                                    :rem   7
230 REM*******INIZIALIZZAZIONE***********
                                    :rem  64
240 SYS 49360                       :rem 155
250 WS=INT(RND(0)*10)+1:WT=INT(RND(0)*225
    )+75:GOSUB530:POKE49155,11-WS
                                    :rem  75
260 POKE49156,11-WS:GETB$:IFB$=""THEN260
                                    :rem  44
270 REM*******SALTO*********************
                                    :rem 194
280 D=PEEK(CD):POKEMB,(PEEK(MB))OR((PEEK(
    MB)AND2)/2):PX=PEEK(XA):PY=PEEK(YA)+2
    1                               :rem  44
290 POKEXP,PX:POKEYP,PY:POKEEN,255:GOSUB8
    80:DX=WS/7:DY=WT/200            :rem 238
300 POKEXP,PX:POKEYP,PY:HT=INT(RND(0)*20)
    +170                            :rem  40
310 PY=PY+DY:IFPY>HTTHENGOTO440     :rem  55
320 PX=PX+DX:IFPX>255THENPX=0:POKEMB,PEEK
    (MB)OR1                         :rem 115
330 IF(PX>80)AND((PEEK(MB)AND1)=1)THENPX=
    10:POKEYP,0:POKEMB,PEEK(MB)AND254
                                    :rem  42
340 TP=PEEK(CD):IF(TPAND1)THENIF(TPAND224
    )THENIFPEEK(YP)<=141THEN560     :rem  11
350 GOTO300                         :rem 100
360 REM*****LETTURA DATI SPRITE**********
                                    :rem  13
370 FORI=SDTOSD+767:READA:POKEI,A:NEXT
                                    :rem 214
380 POKEPR,PA:POKEPR+1,PL:POKEPR+2,CL:POK
    EPR+3,SH:POKEPR+4,SH            :rem 220
390 IFA$="N"THEN410                 :rem  35
400 POKEPR+5,196:POKEPR+6,197:POKEPR+7,19
    8                               :rem   5
410 POKECR,1:POKECR+1,11:POKECR+2,15:POKE
    CR+3,8:POKECR+4,5:POKECR+5,2:rem 197
420 POKECR+6,7:POKECR+7,3:RETURN:rem 247
430 REM******CATTIVO ATTERRAGGIO*********
                                    :rem  68
440 POKEPR,200:GOSUB850:PRINTA$"{HOME}
    { 10 SPAZI}BERSAGLIO MANCATO!
    { 3 SPAZI}"                     :rem 189
450 FORI=1TO1000:NEXT:PRINTA$:POKEEN,254:
    TR=TR-1:SC=SC-10                :rem  85
460 POKEMB,PEEK(MB)AND254:POKEPR,PA:IFTR=
    0THENGOTO480                    :rem 159
470 POKE198,0:GOTO250               :rem 210
480 PRINTA$:GOSUB530:PRINT:PRINT:PRINT:PR
    INTTAB(15)"GAME OVER"           :rem  36
490 PRINT:PRINTTAB(7)"PREMI QUALUNQUE TAS
    TO":POKE198,0                   :rem 196
500 GET B$:IFB$=""THEN500           :rem  79
510 POKEEN,0:POKE53277,0:PRINT"{CLR}":GOS
    UB700:SC=0:TR=10:GOTO210        :rem  49
520 REM******VISUALIZZA PUNTEGGIO********
                                    :rem 139
530 PRINT"{HOME}{WHT}{ 2 SPAZI}PUNTI"SC"
    {SIN} "TAB(13)"UOMINI"TR"{SIN} "TAB(2
    3)"FV"WS"{SIN} ";               :rem 227
540 PRINTTAB(30)"PS"WT"{SIN}":RETURN
                                    :rem  62
550 REM*******BUON ATTERRAGGIO***********
                                    :rem 223
560 POKE 2040,199:FORI=1TO500:NEXTI:GOSUB
    780                             :rem 138
570 PRINTA$"{HOME}{ 10 SPAZI}MISSIONE COM
    PIUTA!{ 9 SPAZI}":FORI=1TO1000:NEXT:P
    RINTA$                          :rem 176
580 IF(TPAND32)=32THENSC=SC+25:GOTO610
                                    :rem  47
590 IF(TPAND64)=64THENSC=SC+50:GOTO610
                                    :rem  56
600 IF(TPAND128)=128THENSC=SC+75:rem 143
610 POKEMB,PEEK(MB)AND254:POKEEN,254:POKE
    2040,PA:POKE198,0:GOTO250       :rem  20
620 REM****AZZERA REGISTRI SONORI********
                                    :rem 132
630 FORI=STOS+24:POKEI,0:NEXT:RETURN
                                    :rem 129
640 REM*****SFONDO DI GIOCO**************
                                    :rem  99
650 RW=1584:CR=54272:FORI=RWTORW+39:POKEC
    R+I,5:NEXT:POKE53280,0:POKE53285,0
                                    :rem 117
660 B$="[<7>]{RVS}{ 40 SPAZI}"      :rem  47
670 FORI=1TO14:PRINTB$;:NEXT:RETURN
                                    :rem  62
680 REM*******TITOLO INIZIALE************
                                    :rem 208
690 PRINT"{CLR}":FORA=0TO10:READL:GOSUB91
    0:NEXT                          :rem  69
700 PRINT"{ 12 GIU'}"TAB(14)"{CYN}(N)OVIZ
    IO":PRINTTAB(14)"{YEL}(I)NTERMEDIO"
                                    :rem 154
710 PRINTTAB(14)"{GRN}(E)SPERTO":POKE198,

720 GETB$:IFB$=""THEN720            :rem  87
730 PRINT"{CLR}"                    :rem 254
740 IFB$="N"THENPOKE53277,224:POKE2045,20
    1:POKE2046,202:POKE2047,203:RETURN
                                    :rem 130
750 IFB$="E"THENPOKE2045,201:POKE2046,202
    :POKE2047,203:RETURN            :rem  69
760 POKE2045,196:POKE2046,197:POKE2047,19
    8:RETURN                        :rem 131
770 REM********SUONO*******************
                                    :rem 216
780 POKES,97:POKES+1,8:POKES+5,0:POKES+6,
    240:POKES+24,15:POKES+4,33      :rem  28
790 FORI=1TO75:NEXT:POKES+4,32:POKES,143:
    POKES+1,10:POKES+4,33           :rem  68
800 FORI=1TO75:NEXT:POKES+4,32:POKES+1,12
    :POKES+4,33:FORI=1TO75:NEXT     :rem  30
810 POKES+4,32:POKES,195:POKES+1,16:POKES
    +4,33:FORI=1TO150:NEXT:POKES+4,32
                                    :rem  32
820 POKES,143:POKES+1,12:POKES+4,33:FORI=
```

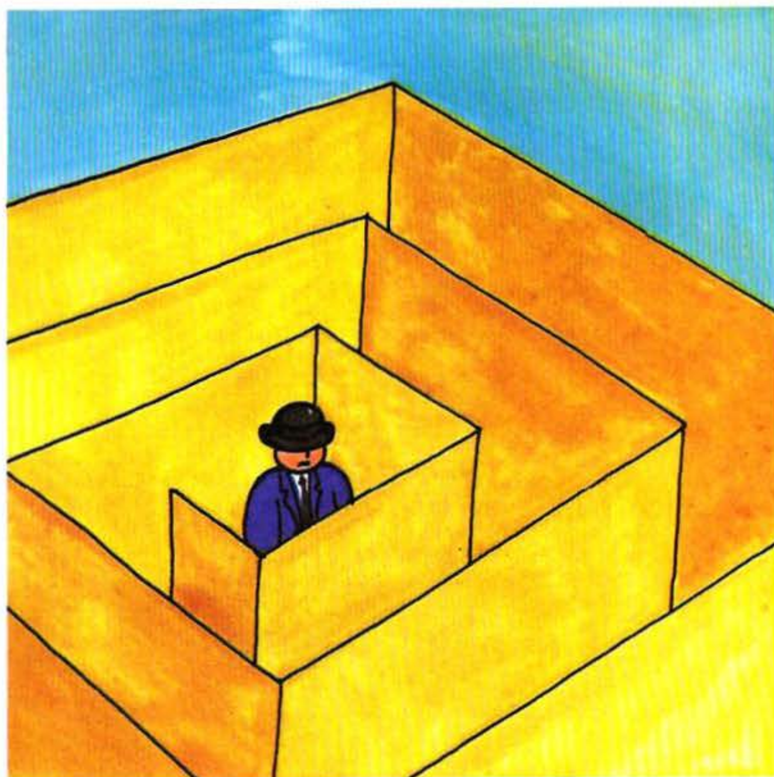
```
    1TO75:NEXT:POKES+4,32:POKES,195
                               :rem  199
830 POKES+1,16:POKES+4,33:FORI=1TO150:NEX
    T:POKES+4,32:RETURN        :rem    9
840 REM*********SPLASH*******************
                               :rem  227
850 POKES,0:POKES+1,64:POKES+5,17:POKES+6
    ,249:POKES+24,15           :rem  160
860 POKES+4,129:FORI=1TO100:NEXT:POKES+4,
    128:FORI=1TO500:NEXT:POKES+1,0:RETURN
                               :rem  197
870 REM*********POOF!********************
                               :rem  154
880 POKES,0:POKES+1,5:POKES+5,145:POKES+6
    ,245:POKES+24,15:POKES+4,129:rem  127
890 FORI=1TO25:NEXT:POKES+4,128:FORI=1TO2
    00:NEXT:POKES+1,0:RETURN   :rem  184
900 REM*****CARATTERI TITOLO*************
                               :rem  235
910 FORI=1038+ATO1478+ASTEP40:T=I+54272:P
    OKET,1:POKET-40,6:POKEI,L  :rem  249
920 FORW=1TO10:NEXTW:NEXTI:RETURN
                               :rem  247
930 I=49152                    :rem   39
940 READ A:IFA=256THENRETURN   :rem  237
950 POKEI,A:I=I+1:GOTO940      :rem  248
960 DATA 0,0,0,0,0,0           :rem  181
970 DATA 0,20,10,88,1,32       :rem  143
980 DATA 173,192,173,2,208,56  :rem  160
990 DATA 233,1,144,38,141,2    :rem   45
1000 DATA 208,173,16,208,41,2  :rem  132
1010 DATA 208,39,173,2,208,205 :rem  188
1020 DATA 10,192,176,31,32,196 :rem  189
1030 DATA 192,173,9,192,141,2  :rem  142
1040 DATA 208,173,16,208,9,2   :rem   92
1050 DATA 141,16,208,76,71,192 :rem  196
1060 DATA 141,2,208,173,16,208 :rem  187
1070 DATA 41,253,141,16,208,206 :rem  236
1080 DATA 3,192,208,94,173,4   :rem  100
1090 DATA 192,141,3,192,169,2  :rem  147
1100 DATA 141,0,192,14,0,192   :rem   75
1110 DATA 172,0,192,170,169,1  :rem  136
1120 DATA 10,202,208,252,141,1 :rem  169
1130 DATA 192,185,0,208,24,105 :rem  187
1140 DATA 1,153,0,208,176,36   :rem   87
1150 DATA 173,16,208,45,1,192  :rem  143
1160 DATA 240,37,185,0,208,205 :rem  189
1170 DATA 9,192,144,29,32,187  :rem  157
1180 DATA 192,153,0,208,173,1  :rem  139
1190 DATA 192,73,255,45,16,208 :rem  205
1200 DATA 141,16,208,76,159,192 :rem  248
1210 DATA 173,16,208,13,1,192  :rem  135
1220 DATA 141,16,208,173,0,192 :rem  185
1230 DATA 74,168,200,152,192,5 :rem  193
1240 DATA 208,170,76,49,234,169 :rem    2
1250 DATA 255,141,15,212,169,128:rem   38
1260 DATA 141,18,212,173,27,212 :rem  236
1270 DATA 96,32,183,192,41,15  :rem  151
1280 DATA 153,0,208,96,32,183  :rem  148
1290 DATA 192,41,40,24,105,50  :rem  136
1300 DATA 141,3,208,96,120,169 :rem  191
1310 DATA 11,141,20,3,169,192  :rem  132
1320 DATA 141,21,3,88,96,120   :rem   90
1330 DATA 169,49,141,20,3,169,256
                               :rem   94
1340 REM*********PARA*******************
                               :rem  146
1350 DATA 0,60,0,1,255,128,7,255:rem   24
1360 DATA 224,15,255,240,31,255,248,63
                               :rem   79
1370 DATA 255,252,63,255,252,59,189,220
                               :rem  144
1380 DATA 049,24,140,16,0,8,8,24:rem   31
1390 DATA 16,4,60,32,3,60,192,1 :rem  231
1400 DATA 153,128,0,255,0,0,60,0:rem    9
1410 DATA 0,60,0,0,60,0,0,36   :rem   57
1420 DATA 0,0,36,0,0,102,0,255 :rem  157
1430 DATA 0,0,0,0,0,0,0,0      :rem  150
1440 DATA 0,0,0,0,0,0,0,0      :rem  151
1450 DATA 0,0,0,0,0,0,0,0      :rem  152
1460 DATA 0,0,0,0,0,3,1,224    :rem    5
1470 DATA 7,66,16,15,79,255,255,127
                               :rem  204
1480 DATA 255,255,64,0,0,64,0,0 :rem  231
1490 DATA 0,0,0,0,0,0,0,0      :rem  156
1500 DATA 0,0,0,0,0,0,0,190    :rem  254
1510 DATA 0,0,0,0,0,0,0,0      :rem  149
1520 DATA 0,0,0,0,0,0,0,0      :rem  150
1530 DATA 0,0,0,0,0,0,0,0      :rem  151
1540 DATA 0,0,0,0,0,0,0,7      :rem  159
1550 DATA 128,0,15,240,0,31,252,0
                               :rem   61
1560 DATA 31,254,0,63,255,0,255,255
                               :rem  182
1570 DATA 1,255,255,7,255,254,31,255
                               :rem  242
1580 DATA 248,255,255,192,0,0,0,0
                               :rem   78
1590 DATA 0,6,0,0,6,0,0,15     :rem  223
1600 DATA 0,0,31,128,0,22,128,0 :rem  211
1610 DATA 038,192,0,38,64,0,102,64
                               :rem  127
1620 DATA 0,230,96,3,230,96,3,230
                               :rem   74
1630 DATA 96,7,230,112,31,246,112,32
                               :rem  224
1640 DATA 30,120,127,254,252,0,6,140
                               :rem  216
1650 DATA 0,7,6,255,255,255,255,255
                               :rem  195
1660 DATA 248,255,255,224,255,255,128,0
                               :rem  140
1670 DATA 21,85,84,26,149,84,21,149
                               :rem  203
1680 DATA 84,21,149,84,26,149,84,25
                               :rem  207
1690 DATA 85,84,25,86,164,25,86,84
                               :rem  168
1700 DATA 26,150,84,21,86,164,21,85
                               :rem  188
1710 DATA 100,21,85,100,21,85,100,21
                               :rem  205
1720 DATA 86,164,21,85,84,0,0,0 :rem  237
1730 DATA 0,0,0,0,0,0,0,0      :rem  153
1740 DATA 0,0,0,0,0,0,0,255    :rem    6
1750 DATA 5,85,80,6,149,80,6,85 :rem    4
1760 DATA 80,6,85,80,6,149,80,5 :rem    0
1770 DATA 149,80,5,154,144,5,153,144
                               :rem  242
1780 DATA 6,153,144,5,89,144,5,89
                               :rem  107
1790 DATA 144,5,89,144,5,89,144,5
                               :rem  107
1800 DATA 90,144,5,85,80,0,0,0 :rem  179
1810 DATA 0,0,0,0,0,0,0,0      :rem  152
1820 DATA 0,0,0,0,0,0,0,255    :rem    5
1830 DATA 1,85,64,1,165,64,1,101:rem   28
1840 DATA 64,1,101,64,1,101,64,1:rem   16
1850 DATA 101,64,1,101,64,1,85,64
                               :rem   77
1860 DATA 1,90,64,1,89,64,1,90 :rem  199
1870 DATA 64,1,86,64,1,86,64,1 :rem  203
1880 DATA 90,64,1,85,64,0,0,0  :rem  138
1890 DATA 0,0,0,0,0,0,0,0      :rem  160
```

```
1900 DATA 0,0,0,0,0,0,0,255        :rem   4
1910 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                                   :rem 193
1920 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                                   :rem 194
1930 DATA 24,0,1,60,128,1,60,128:rem  21
1940 DATA 1,24,128,1,255,128,0,60
                                   :rem  76
1950 DATA 0,0,60,0,0,60,0,0        :rem   9
1960 DATA 36,0,0,36,0,0,36,0       :rem  73
1970 DATA 0,102,0,255              :rem 254
1980 DATA 0,0,0,0,0,0,0,0          :rem 160
1990 DATA 0,0,0,0,0,0,0,0          :rem 161
2000 DATA 0,0,0,0,0,0,0,0          :rem 144
2010 DATA 0,0,0,0,24,0,112,24      :rem  97
2020 DATA 14,204,24,51,6,24,96,3:rem  23
2030 DATA 60,192,1,189,128,112,189,14
                                   :rem  30
2040 DATA 220,255,59,7,126,224,1,255
                                   :rem 231
2050 DATA 128,0,255,0,0,126,0,0 :rem 213
2060 DATA 5,85,80,6,165,80,5,101:rem  33
2070 DATA 80,6,165,80,6,85,80,6 :rem 250
2080 DATA 85,80,6,165,80,5,85,80:rem  49
2090 DATA 5,90,144,5,89,80,5,90 :rem 252
2100 DATA 144,5,85,144,5,85,144,5
                                   :rem  85
2110 DATA 90,144,5,85,80,0,0,0     :rem 174
2120 DATA 0,0,0,0,0,0,0,0          :rem 147
2130 DATA 0,0,0,0,0,0,0,255        :rem   0
2140 DATA 1,85,64,1,169,64,1,149:rem  39
2150 DATA 64,1,169,64,1,89,64,1 :rem 248
2160 DATA 89,64,1,169,64,1,85,64:rem  53
2170 DATA 1,106,64,1,102,64,1,102
                                   :rem  60
2180 DATA 64,1,102,64,1,102,64,1:rem  16
2190 DATA 106,64,1,85,64,0,0,0     :rem 179
2200 DATA 0,0,0,0,0,0,0,0          :rem 146
2210 DATA 0,0,0,0,0,0,0,255        :rem 255
2220 DATA 0,85,0,0,105,0,0,89      :rem 120
2230 DATA 0,0,89,0,0,89,0,0        :rem  23
2240 DATA 89,0,0,89,0,0,85,0       :rem  85
2250 DATA 0,105,0,0,101,0,0,105 :rem 197
2260 DATA 0,0,89,0,0,89,0,0        :rem  26
2270 DATA 105,0,0,85,0,0,0,0       :rem  60
2280 DATA 0,0,0,0,0,0,0,0          :rem 154
2290 DATA 0,0,0,0,0,0,0,255        :rem   7
2300 DATA 0,70,80,220,210,160,160,160
                                   :rem 254
2310 DATA 16,1,18,1,20,18,15,15,16,5,18
                                   :rem 102
```



Trappola!

di **J. Rhees**
trad. ed adatt.
di **M. Cristuib Grizzi**



Costruite dei muri attorno al vostro avversario, cercando di intrappolarlo senza scampo!

Il muro che costruite può svilupparsi verticalmente od orizzontalmente, e non può mai venire in contatto con i muri sul bordo dello schermo, con se stesso o con i muri costruiti dal vostro avversario. Non è nemmeno possibile entrare in contatto con eventuali barriere e ostacoli che possono essere distribuiti casualmente nell'area di costruzione.

Si guadagnano punti cercando di "chiudere" l'avversario, in modo che sia costretto a cozzare contro un altro muro o contro se stesso; in questo caso il vincitore guadagna il round ed un numero di punti in funzione del tempo di durata del round stesso. Il primo giocatore che raggiunge i 100 punti vince la partita.

Le opzioni disponibili nell'inizializzazione del gioco sono: uno o due giocatori, utilizzo del joystick o della tastiera, l'aggiunta di ostacoli al campo di gioco e la possibilità di variare a proprio gradimento la velocità del gioco.

Circa il trenta per cento del programma è scritto in linguaggio macchina, mentre la parte sonora, di temporizzazione e di gestione del punteggio è scritta in BASIC.

Il programma dà il meglio di se stesso se giocato da due giocatori, anche se è possibile giocare da soli per ottenere un buon livello di pratica ed abilità. Si lotta contro il tempo, cercando di sopravvivere il più a lungo possibile e, se si gioca da soli, si dispone di dieci round da superare con il massimo punteggio possibile.

Note alla versione per Vic 20

Il programma per VIC 20, utilizzabile sensa espansioni di memoria, è spezzato in due parti. Occorre sempre caricare il primo programma, il quale provvederà automaticamente a caricare e mandare in esecuzione il secondo.

Commento alla versione per C64

linee	commento
100-120	chiamano le routine per le opzioni ed inizializzano le variabili
130	controlla chi sia il vincitore e passa alla routine di vincita.
140-150	disegnano il bordo dello schermo
160	se si scelgono le barriere, salta alla relativa routine
190	posiziona i giocatori e le direzioni. Le locazioni 251-254 contengono i byte bassi ed alti della posizione di ogni giocatore. Le locazioni 837-838 contengono le direzioni dei due giocatori
200	inizializza il tempo e chiama la routine in linguaggio macchina, che ripassa il controllo al BASIC quando si verifica una collisione. Il punteggio è determinato dall'ammontare del tempo trascorso
210-230	leggono il valore nella locazione 834 per determinare il numero del giocatore entrato in collisione, quindi passano all'apposita routine per aggiornare il punteggio del vincitore
245-250	provocano il lampeggio in corrispondenza della collisione
270-480	permettono la scelta delle opzioni
490-510	pongono casualmente le barriere sullo schermo
520-550	inizializzano i registri sonori e le variabili
560-660	stampano i punteggi ed i totali, quindi ritornano all'inizio del programma
670-1350	inseriscono in memoria la parte in linguaggio macchina

Trappola! versione per VIC20

programma 1

```
100 I=7168:PRINT"{CLR}{ 5 GIU'}
    { 4 SPAZI}ATTENDERE..."        :rem   6
110 READ A:IF A=256 THEN 130       :rem 150
120 POKE I,A:I=I+1:GOTO 110        :rem 226
130 S$=CHR$(131)                   :rem  47
150 FORI=1TOLEN(S$):POKE630+I,ASC(MID$(S$
    ,I)):NEXT:POKE198,I            :rem 124
160 DATA 32,22,28,32,10,29         :rem 241
170 DATA 173,66,3,240,1,96         :rem 254
180 DATA 32,109,29,165,197,208     :rem 208
190 DATA 237,76,15,28,169,130      :rem 157
200 DATA 141,11,144,162,3,181      :rem 127
210 DATA 251,157,102,3,202,16      :rem 127
220 DATA 248,160,100,169,127,141:rem  32
230 DATA 34,145,173,32,145,41      :rem 139
240 DATA 128,74,74,74,74,141       :rem 105
250 DATA 67,3,169,255,141,34       :rem 102
260 DATA 145,173,17,145,74,74      :rem 155
270 DATA 41,7,13,67,3,41           :rem 148
280 DATA 15,201,15,208,3,173       :rem  87
290 DATA 70,3,141,61,3,141         :rem 241
300 DATA 70,3,169,255,141,60       :rem  91
310 DATA 3,165,197,205,62,3        :rem  44
320 DATA 208,4,162,254,208,33      :rem 141
330 DATA 205,63,3,208,4,162        :rem  38
340 DATA 253,208,24,205,64,3       :rem  91
350 DATA 208,4,162,251,208,15      :rem 141
360 DATA 205,65,3,208,4,162        :rem  43
370 DATA 247,208,6,173,60,3        :rem  49
380 DATA 76,139,28,138,45,60       :rem 112
390 DATA 3,41,15,201,15,208        :rem  35
400 DATA 3,173,69,3,141,60         :rem 246
410 DATA 3,141,69,3,136,208        :rem  42
420 DATA 138,140,11,144,206,71     :rem 184
430 DATA 3,208,128,173,72,3        :rem  44
440 DATA 141,71,3,160,0,162        :rem  31
450 DATA 0,173,75,3,240,6          :rem 197
460 DATA 173,61,3,141,60,3         :rem 244
470 DATA 185,60,3,74,176,8         :rem   9
480 DATA 169,22,32,236,28,76       :rem 109
490 DATA 227,28,74,176,8,169       :rem 121
500 DATA 22,32,254,28,76,227       :rem  97
510 DATA 28,74,176,8,169,1         :rem   8
520 DATA 32,236,28,76,227,28       :rem 105
530 DATA 169,1,32,254,28,232       :rem  96
540 DATA 232,200,204,68,3,208      :rem 137
550 DATA 207,96,141,67,3,181       :rem 104
560 DATA 251,56,237,67,3,149       :rem 110
570 DATA 251,181,252,233,0,149     :rem 195
580 DATA 252,96,24,117,251,149     :rem 208
590 DATA 251,181,252,105,0,149     :rem 195
600 DATA 252,96,160,0,173,68       :rem 100
610 DATA 3,201,1,240,35,165        :rem  30
620 DATA 251,197,253,208,29,165    :rem   2
630 DATA 252,197,254,208,23,32     :rem 200
640 DATA 166,29,16,9,169,1         :rem   9
650 DATA 141,66,3,32,120,29        :rem  41
660 DATA 96,169,2,141,66,3         :rem   8
670 DATA 32,142,29,96,160,0        :rem  48
680 DATA 140,66,3,32,166,29        :rem  53
690 DATA 16,7,32,78,29,32          :rem 215
700 DATA 90,29,96,32,90,29         :rem   9
710 DATA 32,78,29,96,177,251       :rem 115
720 DATA 201,32,240,5,169,1        :rem  35
730 DATA 141,66,3,96,173,68        :rem  61
740 DATA 3,201,1,240,11,177        :rem  31
750 DATA 253,201,32,240,5,169      :rem 143
760 DATA 2,141,66,3,96,32          :rem 206
770 DATA 142,29,173,68,3,201       :rem 101
780 DATA 2,240,1,96,169,219        :rem  57
790 DATA 145,253,165,253,24,105 :rem 252
800 DATA 0,133,106,165,254,105     :rem 185
810 DATA 120,133,107,169,6,145     :rem 193
820 DATA 106,96,160,0,169,214      :rem 148
830 DATA 145,251,165,251,24,105 :rem 243
840 DATA 0,133,106,165,252,105     :rem 187
850 DATA 120,133,107,169,2,145     :rem 193
860 DATA 106,96,173,74,3,10        :rem  52
870 DATA 10,56,109,74,3,141        :rem  47
880 DATA 74,3,96,256               :rem 232
```

programma 2

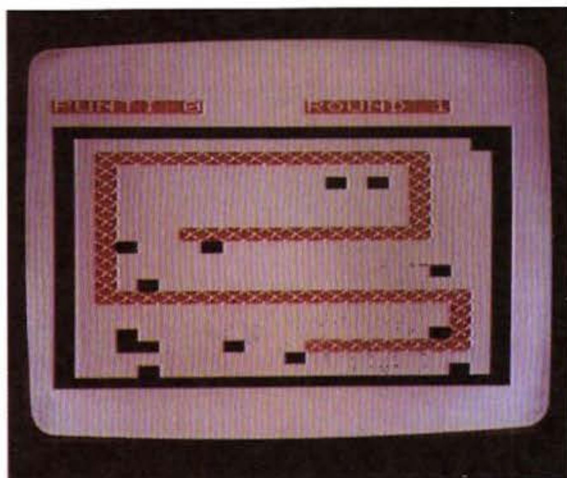
```
100 POKE52,28:POKE56,28:CLR:POKE36878,15:
    GOSUB270:GOSUB480              :rem 246
110 S1=0:S2=0:GOSUB430:IFFLTHEN100
                                   :rem  27
120 PRINT"{CLR}":C=30720           :rem 173
130 R=R+1:IFS1>=100OR(S2>=100ANDNP=2)OR(R
    =11ANDNP=1)THEN540             :rem  23
140 FORA=7724TO7745:POKEA,160:POKEA+C,0:P
    OKEA+440,160:POKEA+440+C,0:NEXT
                                   :rem   9
150 FORA=7746TO8142STEP22:POKEA,160:POKEA
    +C,0:POKEA+21,160:POKEA+21+C,0:NEXT
                                   :rem  60
160 IFB$="S"THENGOSUB510           :rem 165
170 IFNP=2THENPRINT"{HOME}{RVS}{RED}ROSSO
    "S1"{OFF}{ 7 SPAZI}{RVS}{BLU}BLU"S2
                                   :rem 254
180 IFNP=1THENPRINT"{HOME}{RVS}{RED}PUNTI
    "S2"{HOME}"SPC(12)"ROUND"R     :rem 153
190 POKE251,225:POKE252,30:POKE253,235:PO
    KE254,30:POKE837,7:POKE838,D1
                                   :rem  46
200 TI$="000000":SYS7168:SC=INT(TI/60):IF
    NP=1THENSC=SC*LV               :rem 183
210 ONPEEK(834)GOTO220,230         :rem 211
220 SP=PEEK(870)+256*PEEK(871):GOSUB240:S
    2=S2+SC:GOTO120                :rem  41
230 SP=PEEK(872)+256*PEEK(873):GOSUB240:S
    1=S1+SC:GOTO120                :rem  44
240 POKE36877,130:FORA=1TO6        :rem  61
250 POKESP,PEEK(SP)-2*(PEEK(SP)AND128)+12
    8:FORB=1TO400:NEXT:POKE36878,6-A:NEXT
                                   :rem  24
260 POKE36877,0:POKE36878,15:RETURN
                                   :rem  87
270 REM ROUTINE OPZIONI            :rem 203
280 POKE36879,25:PRINT"{CLR}{ 5 GIU'}"TAB
    (7)"{RED}TRAPPOLA!":POKE198,0
                                   :rem  10
290 PRINT"{ 3 GIU'}"TAB(5)"{BLU}{RVS}1
    {OFF} UN GIOCATORE":PRINT"{ 2 GIU'}"T
    AB(5)"{RVS}2{OFF} DUE GIOCATORI"
                                   :rem 151
300 PRINTSPC(5)"{ 2 GIU'}{RVS}3{OFF} FINE
    "                              :rem 235
```

LOAD



Trappola!

```
310 GETA$:IFA$<"1"ORA$>"3"THEN310
                                :rem  54
320 IFA$="3"THENPRINT"{CLR}":END:rem 225
330 NP=VAL(A$):POKE836,NP     :rem 222
340 D1=11:POKE843,0:IFNP=2THEN380
                                :rem 175
350 PRINT"{ 2 GIU'}{RVS}J{OFF}OYSTICK O
    {RVS}T{OFF}ASTIERA"        :rem 239
360 GETA$:IFA$="J"THENPOKE843,1:D1=7:RETU
    RN                          :rem 110
370 IFA$<>"T"THEN360           :rem 104
380 PRINT"{CLR}{ 3 GIU'}CHE TASTO PER SU?
    ":WAIT198,1:A(1)=PEEK(197):POKE198,0
                                :rem 129
390 PRINT"{GIU'}CHE TASTO PER GIU'?":WAIT
    198,1:A(2)=PEEK(197):POKE198,0
                                :rem  50
400 PRINT"{GIU'}CHE TASTO PER SIN.?":WAIT
    198,1:A(3)=PEEK(197):POKE198,0
                                :rem  55
410 PRINT"{GIU'}CHE TASTO PER DES.?":WAIT
    198,1:A(4)=PEEK(197):POKE198,0
                                :rem  43
420 FORA=1TO4:POKE829+A,A(A):NEXT:RETURN
                                :rem  13
430 PRINT"{CLR}{ 5 GIU'}{ 3 SPAZI}VELOCIT
    A'?{ 2 SPAZI}(0-9)"        :rem 244
440 PRINT"{ 2 GIU'}{ 2 SPAZI}O (C) PER OP
    ZIONI"                      :rem 125
450 GETA$:IF(A$<"0"ORA$>"9")ANDA$<>"C"THE
    N450                        :rem 207
460 IFA$="C"THENFL=1           :rem 129
470 LV=VAL(A$):P=60-LV*6:POKE839,P:POKE84
    0,P:LV=LV+1:RETURN         :rem 170
480 PRINT"{CLR}{ 4 GIU'}VUOI LE BARRIERE?
    "                           :rem 163
490 GETB$:IFB$<>"S"ANDB$<>"N"THEN490
                                :rem  52
500 RETURN                      :rem 117
510 FORA=1TO15                  :rem  53
520 Q=RND(1)*430+7746:IFPEEK(Q)<>32OR(Q>7
    899ANDQ<7922)THEN520       :rem  26
530 POKEQ,160:POKEQ+C,0:NEXT:RETURN
                                :rem 242
540 IFNP=2THEN580              :rem   1
550 PRINT"{CLR}{ 8 GIU'}"SPC(7)"{BLU}PUNT
    I:"S2                       :rem 166
560 IFS2>HITHENHI=S2           :rem   0
570 PRINTSPC(8)"{GIU'}{RED}MAX.:"HI:GOTO6
    20                          :rem 243
580 W=-(S1>=100)-2*(S2>=100):PRINT"{CLR}
    { 5 GIU'}"SPC(5)"{RED}GIOC. "W"VINCE!
    "                           :rem 234
590 PRINT"{ 2 GIU'}{BLU}PT{ 2 SPAZI}:"S1:
    PRINTSPC(12)"{SU}PT{ 2 SPAZI}:"S2
                                :rem 171
600 WI(W)=WI(W)+1:PRINT"{ 2 GIU'}VINCE:"W
    I(1):PRINTSPC(12)"{SU}VINCE:"WI(2)
                                :rem 197
610 T1=T1+S1:T2=T2+S2:PRINT"{ 2 GIU'}TOT
    :"T1:PRINTSPC(12)"{SU}TOT :"T2
                                :rem 144
620 PRINTSPC(6)"{ 2 GIU'}PREMI UN TASTO":
    POKE198,0                   :rem  10
630 GETA$:IFA$=""THEN630       :rem  85
640 ONNPGOTO100,110            :rem  93
```



Trappola! versione per C64

```
100 CLR:GOSUB670:GOSUB520:GOSUB540:GOSUB2
    60:GOSUB460                 :rem  19
110 S1=0:S2=0:GOSUB410:IFFLTHEN100
                                :rem  25
120 PRINT"{CLR}":C=54272       :rem 181
130 R=R+1:IFS1>=100OR(S2>=100ANDNP=2)OR(R
    =11ANDNP=1)THEN560         :rem  25
140 FORA=1104TO1143:POKEA,160:POKEA+C,0:P
    OKEA+880,160:POKEA+880+C,0:NEXT
                                :rem 253
150 FORA=1144TO1944STEP40:POKEA,160:POKEA
    +C,0:POKEA+39,160:POKEA+39+C,0:NEXT
                                :rem  67
160 IFB$="S"THENGOSUB490       :rem 172
170 IFNP=2THENPRINT"{HOME}{ 7 SPAZI}{RVS}
    {RED}ROSSO"S1"{OFF}{ 11 SPAZI}{RVS}
    {BLU}BLU"S2                 :rem 254
180 IFNP=1THENPRINT"{HOME}"TAB(8)"{RVS}
    {RED}PUNTI"S2;SPC(8)"ROUND"R:rem 246
190 POKE251,194:POKE252,5:POKE253,214:POK
    E254,5:POKE837,7:POKE838,11 :rem 193
200 TI$="000000":SYS49152:SC=INT(TI/60):I
    FNP=1THENSC=SC*LV          :rem 230
210 ONPEEK(834)GOTO220,230     :rem 211
220 SP=PEEK(870)+256*PEEK(871):GOSUB245:S
    2=S2+SC:GOTO120            :rem  46
230 SP=PEEK(872)+256*PEEK(873):GOSUB245:S
    1=S1+SC:GOTO120            :rem  49
245 FORA=1TO6:POKESP,PEEK(SP)-2*(PEEK(SP)
    AND128)+128:FORB=1TO400:NEXT:rem   3
250 NEXT:RETURN                 :rem 240
260 REM ROUTINE OPZIONI*****************
                                :rem 190
```

```
270 POKE53281,1:PRINT"{CLR}{ 5 GIU'}"TAB(
    15)"{RED}TRAPPOLA!":POKE198,0
                                :rem  244
280 PRINT"{ 3 GIU'}"TAB(13)"{BLU}{RVS}1
    {OFF} UN GIOCATORE":PRINT"{ 2 GIU'}"T
    AB(13)"{RVS}2{OFF} DUE GIOCATORI"
                                :rem  244
290 PRINTSPC(13)"{ 2 GIU'}{RVS}3{OFF} FIN
    E"                          :rem   34
300 GETA$:IFA$<"1"ORA$>"3"THEN300
                                :rem   52
310 IFA$="3"THENPRINT"{CLR}":END:rem  224
320 NP=VAL(A$):POKE836,NP:PRINT"
    { 3 GIU'}{BLK}"TAB(14)"{RVS}J{OFF}OYS
    TICK O"                     :rem  223
330 PRINT"{GIU'}"TAB(8)"{RVS}T{OFF}ASTIER
    A E JOYSTICK 2"             :rem    4
340 GETA$:IFA$="J"THENRETURN    :rem  228
350 IFA$<>"T"THEN340            :rem  100
360 PRINT"{CLR}{ 3 GIU'}{ 7 SPAZI}CHE TAS
    TO PER SU?":WAIT198,1:A(1)=PEEK(197):
    POKE198,0                   :rem  127
370 PRINT"{GIU'}{ 6 SPAZI}CHE TASTO PER G
    IU'?":WAIT198,1:A(2)=PEEK(197):POKE19
    8,0                         :rem   48
380 PRINT"{GIU'}{ 7 SPAZI}CHE TASTO PER S
    IN.?":WAIT198,1:A(3)=PEEK(197):POKE19
    8,0                         :rem   62
390 PRINT"{GIU'}{ 7 SPAZI}CHE TASTO PER D
    ES.?":WAIT198,1:A(4)=PEEK(197):POKE19
    8,0                         :rem   50
400 FORA=1TO4:POKE829+A,A(A):NEXT:RETURN
                                :rem   11
410 PRINT"{CLR}{ 5 GIU'}"SPC(11)"VELOCITA
    ' (0-9)?"                   :rem  207
420 PRINTSPC(7)"{ 2 GIU'}O (C) PER CAMBIA
    RE LE OPZIONI"              :rem  174
430 GETA$:IF(A$<"0"ORA$>"9")ANDA$<>"C"THE
    N430                        :rem  203
440 IFA$="C"THENFL=1            :rem  127
450 LV=VAL(A$):P=60-LV*6:POKE839,P:POKE84
    0,P:LV=LV+1:RETURN          :rem  168
460 PRINT"{CLR}"SPC(8)"{ 4 GIU'}VUOI LE B
    ARRIERE?"                   :rem   84
470 GETB$:IFB$<>"S"ANDB$<>"N"THEN470
                                :rem   48
480 RETURN                      :rem  124
490 FORA=1TO30                  :rem   57
500 Q=RND(1)*870+1104:IFPEEK(Q)<>32OR(Q>1
    463ANDQ<1503)THEN500        :rem  238
510 POKEQ,160:POKEQ+C,0:NEXT:RETURN
                                :rem  240
520 REM INIZIALIZZAZIONE*****************
                                :rem   24
530 FORA=54272TO54296:POKEA,0:NEXT:RETURN
                                :rem   71
540 POKE54287,255:POKE54290,129:POKE54273
    ,7:POKE54296,15:POKE54277,21:rem  166
550 POKE54278,240:RETURN        :rem  175
560 GOSUB520:IFNP=2THEN600      :rem   77
570 PRINT"{CLR}{ 10 GIU'}"SPC(15)"{BLU}PU
    NTI:"S2                     :rem  249
580 IFS2>HITHENHI=S2            :rem    2
590 PRINTSPC(16)"{GIU'}{RED}MAX.:"HI:GOTO
    640                         :rem   38
600 W=-(S1>=100)-2*(S2>=100):PRINT"{CLR}
    { 6 GIU'}"SPC(13)"{RED}GIOC."W"VINCE!
    "                           :rem   35
610 PRINT"{ 2 GIU'}{BLU}{ 4 SPAZI}GIOC. 1
    :"S1:PRINTSPC(25)"{SU}GIOC. 2:"S2
                                :rem   99
```

LOAD

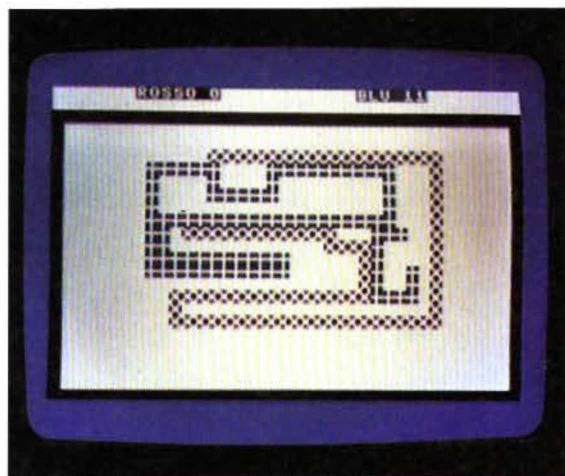
Trappola!

```
620 WI(W)=WI(W)+1:PRINT"{ 2 GIU'}
    { 4 SPAZI}VINCE{ 2 SPAZI}:"WI(1):PRIN
    TSPC(25)"{SU}VINCE{ 2 SPAZI}:"WI(2)
                                :rem  203
630 T1=T1+S1:T2=T2+S2:PRINT"{ 2 GIU'}
    { 4 SPAZI}TOTALE :"T1:PRINTSPC(25)"
    {SU}TOTALE :"T2             :rem   58
640 PRINTSPC(15)"{ 2 GIU'}PREMI UN TASTO"
    :POKE198,0                  :rem   60
650 GETA$:IFA$=""THEN650        :rem   89
660 ONNPGOTO100,110             :rem   95
670 I=49152:IFPEEK(I)=32THENRETURN
                                :rem   97
680 PRINT"{CLR}{ 5 GIU'}"SPC(13)"ATTENDER
    E..."                       :rem   92
690 READ A:IF A=256 THEN RETURN :rem  239
700 POKEI,A:I=I+1:GOTO690       :rem  243
710 DATA 32,22,192,32,229,192   :rem  145
720 DATA 173,66,3,240,1,96      :rem  255
730 DATA 32,72,193,165,197,208  :rem  210
740 DATA 237,76,15,192,169,33   :rem  162
750 DATA 141,4,212,162,3,181    :rem   87
760 DATA 251,157,102,3,202,16   :rem  137
770 DATA 248,160,100,173,0,220  :rem  185
780 DATA 41,15,201,15,208,3     :rem   38
790 DATA 173,70,3,141,61,3      :rem  251
800 DATA 141,70,3,173,1,220     :rem   30
810 DATA 141,60,3,165,197,205   :rem  146
820 DATA 62,3,208,4,162,254     :rem   45
830 DATA 208,33,205,63,3,208    :rem   94
840 DATA 4,162,253,208,24,205   :rem  144
850 DATA 64,3,208,4,162,251     :rem   47
860 DATA 208,15,205,65,3,208    :rem   99
870 DATA 4,162,247,208,6,173    :rem  106
880 DATA 60,3,76,111,192,138    :rem  103
890 DATA 45,60,3,41,15,201      :rem  247
900 DATA 15,208,3,173,69,3      :rem  255
910 DATA 141,60,3,141,69,3      :rem  247
920 DATA 136,208,166,169,32,141 :rem  253
930 DATA 4,212,206,71,3,208     :rem   40
940 DATA 154,173,72,3,141,71    :rem   98
950 DATA 3,160,0,162,0,185      :rem  245
960 DATA 60,3,74,176,8,169      :rem   15
970 DATA 40,32,199,192,76,190   :rem  165
980 DATA 192,74,176,8,169,40    :rem  120
990 DATA 32,217,192,76,190,192  :rem  214
1000 DATA 74,176,8,169,1,32     :rem   46
1010 DATA 199,192,76,190,192,169:rem   58
1020 DATA 1,32,217,192,232,232  :rem  179
1030 DATA 200,204,68,3,208,207  :rem  182
1040 DATA 96,141,67,3,181,251   :rem  146
1050 DATA 56,237,67,3,149,251   :rem  153
1060 DATA 181,252,233,0,149,252 :rem  239
1070 DATA 96,24,117,251,149,251 :rem  250
1080 DATA 181,252,105,0,149,252 :rem  239
1090 DATA 96,160,0,173,68,3     :rem   50
1100 DATA 201,1,240,35,165,251  :rem  174
1110 DATA 197,253,208,29,165,252:rem   46
1120 DATA 197,254,208,23,173,27 :rem  249
```


LOAD

Trappola!

```
1130 DATA 212,16,9,169,1,141     :rem  87
1140 DATA 66,3,32,83,193,96      :rem  55
1150 DATA 169,2,141,66,3,32      :rem  41
1160 DATA 105,193,96,160,0,140   :rem 188
1170 DATA 66,3,173,27,212,16     :rem  94
1180 DATA 7,32,41,193,32,53      :rem  43
1190 DATA 193,96,32,53,193,32    :rem 156
1200 DATA 41,193,96,177,251,201  :rem 244
1210 DATA 32,240,5,169,1,141     :rem  81
1220 DATA 66,3,96,173,68,3       :rem   5
1230 DATA 201,1,240,11,177,253   :rem 177
1240 DATA 201,32,240,5,169,2     :rem  82
1250 DATA 141,66,3,96,32,105     :rem  93
1260 DATA 193,173,68,3,201,2     :rem  93
1270 DATA 240,1,96,169,219,145   :rem 204
1280 DATA 253,165,253,24,105,0   :rem 189
1290 DATA 133,106,165,254,105,212
                                 :rem  82
1300 DATA 133,107,169,6,145,106  :rem 240
1310 DATA 96,160,0,169,214,145   :rem 194
1320 DATA 251,165,251,24,105,0   :rem 180
1330 DATA 133,106,165,252,105,212
                                 :rem  75
1340 DATA 133,107,169,2,145,106  :rem 240
1350 DATA 96,256                 :rem  27
```



Prigione matematica

di **R. Lowe**
trad. ed adatt.
di **M. Cristuib Grizzi**

Sarete in grado di fuggire dalla prigione, trovando la chiave che apre la porta?
Per prima cosa dovrete sconfiggere i mostri matematici che vi sbarrano la strada, ponendovi domande su addizioni, sottrazioni, moltiplicazioni e divisioni.... Un piacevole ed educativo gioco di avventura per bambini, in versione unica per il VIC 20 con almeno 8 Kbyte di espansione e per il C64.



All'inizio di "Prigione Matematica" vi viene chiesto il livello di difficoltà: i livelli uno o due comportano soltanto domande su addizioni e sottrazioni con numeri positivi, mentre il livello tre aggiunge anche la moltiplicazione e la divisione (sempre solo con numeri positivi). Il livello quattro pone tutti i tipi di domande, sia con numeri positivi che con numeri negativi.
Si incomincia avendo davanti a sé un cancello all'esterno di un'antica dimora.
Per prima cosa c'è una strada per oltrepassare il cancello: vi troverete di fronte alla porta di ingresso, che sarà chiusa. Se potrete aprirla, sarete in grado di entrare nell'antica dimora e presto vi troverete nella prigione sotterranea. Qui comincia l'avventura.
Vagabondando per la prigione, potrete trovare molte chiavi e porte diversamente colorate: ovviamente, la chiave di un colore aprirà soltanto una porta dello stesso colore. Una delle tante porte incontrate sarà quella che permette di uscire dalla prigione matematica, provocando il termine del gioco.
Nella vostra ricerca dell'uscita potrete inoltre trovare una mappa: essa può tornare molto utile, in quanto, dopo averla trovata, sarà sufficiente digitare MAPPA per sapere dove vi trovate. La mappa usa R per rappresentare le stanze, H per le vie di passaggio, M per i mostri e * per il punto in cui vi trovate.
Il gioco riconosce un numero limitato di parole. I verbi sono:

prendi	prende un oggetto
inv	inventario
guarda	descrive il luogo
apri	apre una porta
chiudi	chiude una porta
accendi	accende una luce
spegni	spegne una luce
mappa	guarda la mappa
n	va a nord
s	va a sud
e	va a est
o	va a ovest

Mentre i vocaboli possibili sono:

cancello
porta
biglietto
chiave
lampada
cartone del latte
oro
mappa

Se vi trovate in difficoltà, digitando ALGEBRA potrete avere una via d'uscita. Le informazioni circa la prigione sono contenute nella matrice FL (10,10), che viene riempita di stanze, porte e mostri nelle linee 1620-1850. Le linee 1620-

1640 piazzano a caso i mostri, in relazione al livello scelto inizialmente.
Le linee 1650-1680 piazzano a caso le porte nella prigione. Le linee 1690-1710 si assicurano che sia aperta una via a ciascun mostro, mentre le linee 1720-1770 sistemano a caso le porte e le chiavi colorate.
Il gioco funziona sia sul VIC 20 con espansione di memoria da almeno 8 Kbyte che sul C64: la SYS in linea 120 determina su quale tipo di computer è stato caricato il programma ed inizializza la variabile CC in funzione del numero di colonne di schermo. Questa informazione viene in seguito utilizzata per formattare qualsiasi cosa venga visualizzata sullo schermo.

Prigione matematica per VIC 20 e C64

```
110 DIM CM$(22),CM(22),IN$(5),IN(5),OB$(1
    3),OB(13),DS$(40),FL(10,10)  :rem  157
120 SYS65517:CC=PEEK(781)        :rem  173
130 IFCC=40THENPOKE53281,1       :rem    8
150 GOSUB700:GOSUB1620           :rem   48
155 PRINT"{CLR}"                 :rem  255
160 GOSUB790:CM(2)=1:CM(5)=1:FORI=14TO19:
    CM(I)=1:NEXT:OB(1)=1         :rem   79
170 PR$=DS$(1):GOSUB2570:IFGTTHENPR$=DS$(
    2):GOSUB2570                 :rem  188
180 IFGT=0THENPR$=DS$(3):GOSUB2570
                                 :rem   41
190 GOSUB2310:IFPL=2ORPL=5THENGOSUB800:GO
    TO190                        :rem  183
200 GOSUB790:FORI=10TO19:CM(I)=1:NEXTI:CM
    (22)=I                       :rem   96
210 IFPL=16THEN280               :rem   43
220 IFPL>16THENGOTO250           :rem   99
230 GOTO170                      :rem  102
240 PRINT"{CLR}":PR$=DS$(4):GOSUB2570
                                 :rem  255
250 GOSUB790:FORI=14TO19:CM(I)=1:NEXT:GOS
    UB2310:IFPL=16THEN280        :rem  109
260 IFPL=15THENPR$=DS$(4):GOSUB2570
                                 :rem   96
270 GOTO250                      :rem  105
280 GOSUB790:PRINT"{CLR}":CM(1)=1:CM(3)=1
    :CM(6)=1:FORI=14TO19:CM(I)=1:NEXT
                                 :rem  241
290 CM(4)=1                      :rem   30
300 PR$=DS$(5):GOSUB2570:IFMB=0THENPR$=DS
    $(6):GOSUB2570               :rem   32
310 IFMBTHENPR$=DS$(7):CM(7)=1:GOSUB2570:
    IFNT=1THENPR$=DS$(11):GOSUB2570
                                 :rem  112
320 IFFD=0THENPR$=DS$(8):NT=1:GOSUB2570
                                 :rem   99
330 IFFDTHENPR$=DS$(9):NT=0:GOSUB2570
                                 :rem  247
340 GOSUB2310:IFPL=15THEN300     :rem  167
350 IF(PL=18)OR(PL=19)THENGOTO300
                                 :rem  234
360 IFPL=17THENGOTO160           :rem  104
370 IF(PL=16)AND(FD=0)THEN320    :rem  153
380 IFPL=16ANDFD=0THEN320        :rem  248
390 IFPL=16THEN450               :rem   51
400 IFPL=1THENPR$=DS$(10):GOSUB2570
                                 :rem   84
410 IF(PL=3)OR(PL=6)THENGOSUB900:GOTO340
                                 :rem  212
415 IFPL=7ANDNT=0THENPRINT"NON VEDO FOGLI
     QUI!":GOTO340               :rem  191
420 IFPL<>7THEN340               :rem   56
430 PR$="C'E' UNA DOMANDA SUL FOGLIO.":GO
    SUB2570                      :rem  246
435 GOSUB1320:IFWR=1THENGOTO160  :rem  193
440 PR$=DS$(12):GOSUB2570:PR$=DS$(13):GOS
    UB2570:FD=1:MB=0:GOTO340     :rem   57
450 GOSUB790:PRINT"{CLR}{ 7 GIU'}";
                                 :rem    9
470 PR$="UNA BOTOLA SI APRE SOTTO DI TE,
    ED INIZI A SCIVOLARE SU UNA "
                                 :rem  183
480 PR$=PR$+"CASCATA. PASSI UN'INSEGNA AL
     NEON CHE DICE :":GOSUB2570:PRINT
                                 :rem  108
490 PR$="{ 5 SPAZI}'PRIGIONE MATEMATICA'"
    :GOSUB2570                   :rem  237
500 PRINT:PR$="{ 13 SPAZI}'LIVELLO "+STR$
    (LV)+"'":GOSUB2570:PRINT:PRINT
                                 :rem  200
510 PR$="PREMI UN TASTO PER CONTINUARE":G
    OSUB2570                     :rem  239
520 GETA$:IFA$=""THEN520         :rem   81
530 LT=1:PX=INT(RND(1)*10)+1:PY=INT(RND(1
    )*10)+1:FL(PY,PX)=2          :rem    0
540 GOSUB790:CM(9)=1:FORI=12TO19:CM(I)=1:
    NEXT:CM(9)=LT:CM(22)=1       :rem  251
550 IFFL(PY,PX)=2THENPRINT"{CLR}":PR$=DS$
    (14):GOSUB2570               :rem  193
560 IFFL(PY,PX)=1THENPRINT"{CLR}":PR$=DS$
    (15):GOSUB2570               :rem  194
570 IFLT=1THENPR$=DS$(16):GOSUB2570
                                 :rem  102
580 GOSUB2310                    :rem  227
590 IFPL=9THENGOSUB950:GOTO580   :rem  156
600 IFPL=12THENGOSUB970:GOTO580  :rem  192
610 IFPL=13THENGOSUB1000:GOTO580:rem  227
620 IFPL=15ANDLO=0THEN550        :rem   10
630 IFPL=15ANDLO=1THENGOSUB1020:GOTO570
                                 :rem  196
640 IFPL=14THEN580               :rem   51
650 IFPL=22THENGOSUB2470:GOTO670:rem  243
660 IFPL>15THENGOSUB1560:IF(LO=0)OR(WA=1)
    THEN580                      :rem  204
670 ONZGOSUB1860,1860,2000       :rem  204
680 GOTO670                      :rem  116
690 END                          :rem  118
700 PRINT"{CLR}{ 9 GIU'}":IFCC=40THENPRIN
    TTAB(10);:GOTO720            :rem   16
710 PRINTTAB(1);                 :rem  185
720 PRINT" PRIGIONE MATEMATICA{ 2 SPAZI}
    ":PRINT"{GIU'}":IFCC=40THENPRINTTAB(1
    2);:GOTO740                  :rem   57
730 PRINTTAB(4);                 :rem  190
740 PRINT                        :rem   40
750 PRINT"{HOME}{ 21 GIU'}"TAB(8);:PR$="I
    NSERISCI IL LIVELLO (1-5)"   :rem   58
760 GOSUB2570:PRINTTAB(8);:INPUTLV:IFLV<1
    ORLV>5THENGOTO750            :rem   75
770 FORI=1TO22:READCM$(I):NEXT:FORI=1TO5:
    READIN$(I):NEXT              :rem   52
780 FORI=1TO29:READDS$(I):NEXT:RETURN
                                 :rem  137
790 FORI=1TO22:CM(I)=0:NEXTI:IFIN(5)=1THE
    NCM(20)=1:RETURN             :rem   38
800 IFPL=2ANDGT=1THENPR$="IL CANCELLO E'G
    IA'APERTO.":GOSUB2570        :rem   63
810 IFPL=2ANDGT=0THENGT=1:PRINT"VA BENE."
                                 :rem   48
820 IFPL=5ANDGT=0THENPR$="IL CANCELLO E'G
    IA'CHIUSO.":GOSUB2570        :rem   67
830 IFPL=5ANDGT=1THENGT=0:PRINT"VA BENE."
                                 :rem   53
840 RETURN                       :rem  124
850 IFPL=1ANDFD=1THENPR$="LA PORTA E'GIA'
    APERTA.":GOSUB2570           :rem   97
860 IFPL=1ANDFD=0THENFD=1:PRINT"VA BENE."
                                 :rem   18
```

```
870  IFPL=4ANDFD=0THENPR$="LA PORTA E'GIA'
     CHIUSA.":GOSUB2570              :rem  101
880  IFPL=4ANDFD=1THENFD=0:PRINT"VA BENE."
                                     :rem   23
890  RETURN                          :rem  129
900  IFPL=3ANDMB=1THENPR$="IL CARTONE DEL
     LATTE E'GIA'APERTO.":GOSUB2570
                                     :rem   79
910  IFPL=3ANDMB=0THENMB=1:PRINT"VA BENE."
     :CM(7)=1                        :rem  218
920  IFPL=6ANDMB=0THENPR$="IL CARTONE DEL
     LATTE E'GIA'CHIUSO.":GOSUB2570
                                     :rem   83
930  IFPL=6ANDMB=1THENMB=0:PRINT"VA BENE."
     :CM(7)=0                        :rem  222
940  RETURN                          :rem  125
950  IFIN(1)=1THENPR$="HAI GIA' LA LAMPADA
     .":GOSUB2570:RETURN             :rem   65
960  IFLT=1THENLT=0:IN(1)=1:PRINT"VA BENE.
     ":CM(9)=0:RETURN                :rem  254
970  IFIN(1)=0THENPR$="NON HAI LA LAMPADA.
     ":GOSUB2570:RETURN              :rem   53
980  IFLO=0THENPRINT"VA BENE.":LO=1:RETURN
                                     :rem  116
990  IFLO=1THENPR$="LA LAMPADA E'GIA'ACCES
     A.":GOSUB2570:RETURN            :rem    1
1000 IFLO=0THENPR$="LA LAMPADA E'GIA'SPEN
     TA.":GOSUB2570:RETURN           :rem   74
1010 IFLO=1THENPRINT"VA BENE.":LO=0:RETUR
     N                               :rem  149
1020 PRINT"{CLR}"DS$(16-FL(PX,PY))
                                     :rem   39
1030 EC=PX+1:DN$="EST":IFEC=11THENGOTO105
     0                               :rem    8
1040 IFFL(PY,EC)<>0THENGOSUB1120:rem   177
1050 WC=PX-1:DN$="OVEST":IFWC=0THENGOTO10
     70                              :rem  165
1060 IFFL(PY,WC)<>0THENGOSUB1120:rem   197
1070 NC=PY-1:DN$="NORD":IFNC=0THENGOTO109
     0                               :rem   58
1080 IFFL(NC,PX)<>0THENGOSUB1120:rem   189
1090 SC=PY+1:DN$="SUD":IFSC=11THENGOTO111
     0                               :rem   40
1100 IFFL(SC,PX)<>0THENGOSUB1120:rem   187
1110 RETURN                          :rem  163
1120 PRINTDN$" E'UN PASSAGGIO ATTRAVERSO
     UNA PORTA.":RETURN              :rem  181
1130 IFGK=0ANDBK=0ANDRK=0THENPRINT"QUALE
     CHIAVE?":RETURN                 :rem  126
1140 PRINT"VA BENE."                 :rem  118
1150 IFGKTHENGK=0:IN(2)=1:RETURN:rem   242

1160 IFBKTHENBK=0:IN(3)=1:RETURN:rem   234
1170 IFRKTHENRK=0:IN(4)=1:BD=2:RETURN

                                     :rem   59
1180 RETURN                          :rem  170
1190 IFGD<>1THEN1220                 :rem  132
1200 IFIN(2)=0THENPR$=DS$(10):GOSUB2570:R
     ETURN                           :rem   26
1210 IFIN(2)=1THENPR$=DS$(27):GOSUB2570:M
     A=1:CM(11)=1:CM(1)=0:RETURN:rem   254
1220 IFBD<>1THEN1250                 :rem  124
1230 IFIN(3)=0THENPR$=DS$(10):GOSUB2570:R
     ETURN                           :rem   30
1240 IFIN(3)=1THENPR$=DS$(28):GOSUB2570:C
     M(8)=1:RK=1:CM(1)=0:RETURN :rem   232
1250 IFRD<>1THEN1280                 :rem  146
1260 IFIN(4)=0THENPR$=DS$(10):GOSUB2570:R
     ETURN                           :rem   34
1270 IFIN(4)=1THENPRINT"{CLR}":PR$=DS$(29
     ):GOSUB2570:GOSUB2520           :rem   53
1280 RETURN                          :rem  171
```

LOAD

Prigione matematica

```
1290 G=INT(RND(1)*LV*10)+1:PR$="HAI TROVA
     TO "+STR$(G)+" PEZZI D'ORO."
                                     :rem  145
1300 GOSUB2570:GP=GP+G:PR$="ORA POSSIEDI
     "+STR$(GP)+" PEZZI D'ORO.":GOSUB2570
                                     :rem   46
1310 CM(10)=0:RETURN                 :rem  142
1320 OP=INT(RND(1)*4)+1              :rem  254
1340 IF(LV<=2)AND(OP>=3)THEN1320:rem    91
1350 PRINT"COS'E'...";               :rem  215
1360 ONOPGOTO1370,1400,1430,1460:rem   230
1370 X=INT(RND(1)*LV*10):Y=INT(RND(1)*LV*
     10)                             :rem   34
1380 IFLV>4THENX=X-INT(RND(1)*LV*5):Y=Y-I
     NT(RND(1)*LV*5)                 :rem  168
1390 Z1=X+Y:PRINTX" + "Y"?":INPUTZ$:GOSUB
     1510:IFWR=2THEN1390             :rem  129
1395 RETURN                          :rem  178
1400 X=INT(RND(1)*LV*10):Y=INT(RND(1)*LV*
     10)                             :rem   28
1410 IFLV<=3THENIFY>XTHENT=X:X=Y:Y=T
                                     :rem  180
1420 Z1=X-Y:PRINTX" - "Y"?":INPUTZ$:GOSUB
     1510:IFWR=2THEN1420             :rem  121
1425 RETURN                          :rem  172
1430 X=INT(RND(1)*LV*5):Y=INT(RND(1)*LV*5
     )                               :rem  199
1440 IFLV>=4THENX=X-INT(RND(1)*LV*2):Y=Y-
     INT(RND(1)*LV*2)                :rem  220
1450 Z1=X*Y:PRINTX" * "Y"?":INPUTZ$:GOSUB
     1510:IFWR=2THEN1450             :rem  121
1455 RETURN                          :rem  175
1460 X=INT(RND(1)*LV*5):Y=INT(RND(1)*LV*5
     )                               :rem  202
1470 IFLV>=4THENX=X-INT(RND(1)*LV*3):Y=Y-
     INT(RND(1)*LV*3)                :rem  225
1480 IFY=0THENY=1                    :rem   24
1490 Z1=X/Y:IFZ1<>INT(Z1)THEN1460
                                     :rem    5
1500 PRINTX" / "Y"?":INPUTZ$         :rem   66
1505 GOSUB1510:IFWR=2THEN1490        :rem  240
1506 RETURN                          :rem  172
1510 IFVAL(Z$)<>0THENZ2=VAL(Z$):GOTO1540
                                     :rem  153
1520 IFZ$<>"0"THENPR$="RISPONDI PRIMA ALL
     A DOMANDA!":GOSUB2570:WR=2:RETURN
                                     :rem   61
1530 Z2=VAL(Z$)                      :rem   68
1540 IFZ1=Z2THENPRINT"{CLR}GIUSTO!!":WR=0
     :RETURN                         :rem  199
1550 PRINT"{CLR}SBAGLIATO!!":WR=1:RETURN
                                     :rem  114
1560 WA=0:IFLO=0THENPR$="DEVO PRIMA ACCEN
     DERE LA LAMPADA!.":GOSUB2570:GOTO161
     0                               :rem  200
1570 C=ABS(PL=18)-ABS(PL=19):D=ABS(PL=17)
     -ABS(PL=16):TX=PX+C:TY=PY+D:rem   121
1580 IFTX=11ORTX=0ORTY=11ORTY=0THENPRINT"
     C'E'UN MURO QUI.":WA=1:GOTO1610
                                     :rem   94
```

```
1590 IFFL(TY,TX)=0THENPRINT"C'E'UN MURO Q
     UI.":WA=1:GOTO1610            :rem  135
1600 PX=TX:PY=TY                   :rem   37
1610 Z=FL(PY,PX):RETURN            :rem  217
1620 FORI=1TO10:FORJ=1TO10:FL(I,J)=0:NEXT
     J:NEXTI                       :rem   22
1625 NM=INT(RND(1)*10)+LV:I=1      :rem  145
1630 X=INT(RND(1)*10)+1:Y=INT(RND(1)*10)+
     1:IFFL(Y,X)=3THEN1630         :rem   51
1640 FL(Y,X)=3:I=I+1:IFI<=NMTHEN1630
                                   :rem  127
1650 FORI=1TONM*2                  :rem    4
1660 X=INT(RND(1)*10)+1:Y=INT(RND(1)*10)+
     1                             :rem   68
1670 IFFL(Y,X)=3THEN1660           :rem  137
1680 FL(Y,X)=0:NEXTI               :rem  190
1690 FORI=1TO10:FORJ=1TO10         :rem   79
1700 IFFL(I,J)=3THENGOSUB1780      :rem  232
1710 NEXTJ:NEXTI                   :rem   20
1720 GOSUB1840:GX=X:GY=Y:IF(BX=XANDBY=Y)O
     R(RX=XANDRY=Y)ORFL(Y,X)=0THEN1720
                                   :rem   78
1730 GOSUB1840:BX=X:BY=Y:IF(GX=XANDGY=Y)O
     R(RX=XANDRY=Y)ORFL(Y,X)=0THEN1730
                                   :rem   80
1740 GOSUB1840:X1=X:Y1=Y:IF(X2=XANDY2=Y)O
     R(X3=XANDY3=Y)ORFL(Y,X)=0THEN1740
                                   :rem  200
1750 GOSUB1840:X2=X:Y2=Y:IF(X1=XANDY1=Y)O
     R(X3=XANDY3=Y)ORFL(Y,X)=0THEN1750
                                   :rem  202
1760 GOSUB1840:X3=X:Y3=Y:IF(X1=XANDY1=Y)O
     R(X2=XANDY2=Y)ORFL(Y,X)=0THEN1760
                                   :rem  204
1770 RETURN                        :rem  175
1780 FORK=JTO10:GOSUB1800:NEXTK:FORK=JTO1
     STEP-1:GOSUB1800:NEXTK        :rem  124
1790 FORK=ITO10:GOSUB1820:NEXTK:FORK=ITO1
     STEP-1:GOSUB1820:NEXTK:RETURN
                                   :rem  153
1800 IFFL(I,K)=0THENFL(I,K)=INT(RND(1)*2)
     +1                            :rem  209
1810 RETURN                        :rem  170
1820 IFFL(K,J)=0THENFL(K,J)=INT(RND(1)*2)
     +1                            :rem  213
1830 RETURN                        :rem  172
1840 X=INT(RND(1)*10)+1:Y=INT(RND(1)*10)+
     1:IFFL(Y,X)=0THEN1840         :rem   54
1850 RETURN                        :rem  174
1860 GOSUB790:FORI=12TO19:CM(I)=1:NEXT:CM
     (22)=1                        :rem   62
1870 PRINT"{CLR}":PR$=DS$(16-Z):GOSUB2570
     :IFLO=1THENGOSUB1030          :rem   56
1880 GOSUB2270:GOSUB2200           :rem  154
1890 GOSUB2310:IFPL=15ANDLOTHENGOSUB1020:
     GOTO1880                      :rem   68
1900 IFPL=20THENGOSUB2850:GOTO1890
                                   :rem   87
1910 IFPL=15ANDLO=0THEN1870        :rem  115
1920 IFPL=1THENGOSUB1190:GOTO1890
                                   :rem   36
1930 IFPL=14THEN1890               :rem  155
1940 IFPL=11THENGOSUB2490:GOTO1890
                                   :rem   91
1950 IFPL=8THENGOSUB1130:GOTO1890
                                   :rem   40
1960 IFPL=22THENGOSUB2470:RETURN:rem   50
1970 IFPL>15THENGOSUB1560:IF(LO=0)OR(WA=1
     )THEN1890                     :rem   54
1980 RETURN                        :rem  178
2000 GOSUB790:FORI=10TO19:CM(I)=1:NEXTI:C
     M(22)=1                       :rem  120
```

```
2010 PR$=DS$(23):GOSUB2570:PR$=DS$(24):GO
     SUB2570:GOSUB1320:IFWR=0THENGOTO2050
                                   :rem  132
2020 GP=0:PR$=DS$(25):GOSUB2570:FORW=1TO2
     000:NEXTW                     :rem   27
2030 X=INT(RND(1)*10+1):Y=INT(RND(1)*10)+
     1:IFFL(Y,X)=0THEN2030         :rem   38
2040 PX=X:PY=Y:Z=FL(PY,PX):RETURN
                                   :rem  199
2050 PR$=DS$(26):GOSUB2570:FL(PY,PX)=2
                                   :rem  207
2060 PR$=DS$(14):GOSUB2570:IFLOTHENGOSUB1
     030                           :rem  155
2070 GOSUB2270:GOSUB2200           :rem  146
2080 GOSUB2310                     :rem   16
2090 IFPL=20THENGOSUB2850:GOTO2080
                                   :rem   80
2100 IFPL=15ANDLOTHENGOSUB1020:GOTO2070
                                   :rem  173
2110 IFPL=15ANDLO=0THEN2060        :rem  100
2120 IFPL=14THEN2080               :rem  139
2130 IFPL=11THEN2490:GOTO2080      :rem  203
2140 IFPL=8THENGOSUB1130:GOTO2080
                                   :rem   24
2150 IFPL=1THENGOSUB1190:GOTO2080
                                   :rem   24
2160 IFPL=10THENGOSUB1290:GOTO2080
                                   :rem   74
2170 IFPL=22THENGOSUB2470:RETURN:rem   44
2180 IFPL>15THENGOSUB1560:IF(LO=0)OR(WA=1
     )THEN2080                     :rem   40
2190 RETURN                        :rem  172
2200 IF(X1=PXANDY1=PY)ANDGD<>2THENPR$=DS$
     (17):GOSUB2570:GD=1:CM(1)=1:rem  122
2210 IF(X2=PXANDY2=PY)ANDBD<>2THENPR$=DS$
     (19):GOSUB2570:BD=1:CM(1)=1:rem  117
2220 IF(X3=PXANDY3=PY)ANDRD<>2THENPR$=DS$
     (18):GOSUB2570:RD=1:CM(1)=1:rem  151
2230 IF(X1<>PXORY1<>PY)ANDGD<>2THENGD=0
                                   :rem  209
2240 IF(X2<>PXORY2<>PY)ANDBD<>2THENBD=0
                                   :rem  202
2250 IF(X3<>PXORY3<>PY)ANDRD<>2THENRD=0
                                   :rem  237
2260 RETURN                        :rem  170
2270 IFBX=PXANDBY=PYANDIN(3)=0THENPR$=DS$
     (21):GOSUB2570:BK=1:CM(8)=1:rem  167
2280 IFGX=PXANDGY=PYANDIN(2)=0THENPR$=DS$
     (20):GOSUB2570:GK=1:CM(8)=1:rem  181
2290 IFRX=PXANDRY=PYANDIN(4)=0THENPR$=DS$
     (22):GOSUB2570:RK=1:CM(8)=1:rem  219
2300 RETURN                        :rem  165
2310 INPUTC$:PL=0:FORI=1TO22:IFCM$(I)=C$T
     HENPL=I:I=23                  :rem   16
2320 NEXT                          :rem    6
2330 IFPL=0THENPRINT"NON PUOI FARE QUESTO
     !":GOTO2310                   :rem  225
2340 N$="":I=1                     :rem  173
2350 T$=MID$(C$,I,1)               :rem    7
2360 IFASC(T$)=32THENL=I:GOTO2390
                                   :rem  222
2370 IFI=LEN(C$)THENGOTO2410       :rem  167
2380 I=I+1:GOTO2350                :rem   53
2390 N$=MID$(C$,L+1,LEN(C$))       :rem  202
2400 IFCM(PL)=0THENPRINT"NON VEDO "N$" QU
     I!":GOTO2310                  :rem   87
2410 IFC$="INV"THENGOTO2430        :rem   92
2420 RETURN                        :rem  168
2430 PRINT"HAI CON TE :"           :rem   31
2440 FORI=1TO5:IFIN(I)=1THENPRINT"UN/A "I
     N$(I)                         :rem   16
```



Prigione matematica

```
2450 NEXTI:IFGP<>0THENPRINTGP" PEZZI D'OR
     O."                          :rem  15
2460 RETURN                       :rem 172
2470 PX=INT(RND(1)*10)+1:PY=INT(RND(1)*10
     )+1:IFFL(PY,PX)=0THEN2470    :rem 118
2480 Z=FL(PY,PX):RETURN           :rem 223
2490 IFIN(5)=1THENPR$="HAI GIA' LA MAPPA!
     ":GOSUB2570:RETURN           :rem 232
2500 IFMA=0THENPRINT"QUALE MAPPA?":RETURN
                                  :rem 145
2510 PRINT"VA BENE.":IN(5)=1:MA=0:CM(20)=
     1:GD=2:RETURN                :rem 171
2520 PR$="{ 2 GIU'}HAI TROVATO LA VIA D'U
     SCITA DALLA PRIGIONE MATEMATICA.":GO
     SUB2570                      :rem  27
2530 PR$="{GIU'}{ 2 SPAZI}HAI VINTO "+STR
     $(GP)+" PEZZI D'ORO NEL TUO VIAGGIO.
     ":GOSUB2570                  :rem 163
2540 PRINT"{GIU'} GIOCHI ANCORA";:INPUT A
     $                            :rem 126
2550 IFA$="S"THENRUN              :rem 184
2560 END                          :rem 164
2570 IFLEN(PR$)<=CCTHENPRINTPR$:RETURN
                                  :rem 238
2580 T$=MID$(PR$,CC,1):IFASC(T$)=32THENI=
     CC:GOTO2620                  :rem  43
2590 I=CC-1                       :rem  58
2600 T$=MID$(PR$,I,1):IFASC(T$)=32THENGOT
     O2620                        :rem 161
2610 I=I-1:GOTO2600               :rem  49
2620 A$=MID$(PR$,I+1,LEN(PR$)):B$=MID$(PR
     $,1,I-1):PRINTB$             :rem 195
2630 IFLEN(A$)<=CCTHENPRINTA$:RETURN
                                  :rem  41
2640 PR$=A$:GOTO2570              :rem 117
2650 DATAAPRI PORTA,APRI CANCELLO,APRI CA
     RTONE,CHIUDI PORTA           :rem 254
2655 DATACHIUDI CANCELLO,CHIUDI CARTONE
                                  :rem 209
2660 DATAPRENDI FOGLIO,PRENDI CHIAVE,PREN
     DI LAMPADA,PRENDI ORO,PRENDI MAPPA
                                  :rem  33
2665 DATAACCENDI LAMPADA,SPEGNI LAMPADA,I
     NV                           :rem 191
2670 DATAGUARDA,N,S,E,O,MAPPA,MAPPA,ALGEB
     RA,LAMPADA                   :rem 238
2675 DATACHIAVE VERDE,CHIAVE BLU,CHIAVE R
     OSSA,MAPPA                   :rem 210
2680 DATASEI RIVOLTO A NORD.VEDI UN CANCE
     LLO.,IL CANCELLO E'APERTO.   :rem  61
2690 DATAIL CANCELLO E'CHIUSO.,SEI IN UNA
      FITTA FORESTA.              :rem 104
2700 DATASEI DAVANTI AD UNA VECCHIA CASA.
     C'E' UN CARTONE DEL LATTE SULLA PORT
     A.                           :rem 118
2710 DATAIL CARTONE E'CHIUSO.,IL CARTONE
     E'APERTO.,LA PORTA E'CHIUSA.
                                  :rem 178
2720 DATALA PORTA E' APERTA.,LA PORTA E'C
     HIUSA. NON HAI LA CHIAVE GIUSTA.
                                  :rem 218
2730 DATAC'E'UN FOGLIO NEL CARTONE.,LA PO
     RTA SI APRE CON UN BANG!     :rem 247
2740 DATAIL FOGLIO SCOMPARE ED IL CARTONE
      SI RICHIUDE.                :rem 215
2750 DATASEI IN UNA STANZA FREDDA E UMIDA
     ,SEI IN UN LUNGO E STRETTO CORRIDOIO
     .                            :rem 142
2760 DATAC'E' QUI UNA LAMPADA SPENTA.,C'E
     ' QUI UNA PORTA VERDE.       :rem 122
2770 DATAC'E' QUI UNA PORTA ROSSA.,C'E' Q
     UI UNA PORTA BLU.            :rem  59
2780 DATAC'E' QUI UNA CHIAVE VERDE.,C'E'
     QUI UNA CHIAVE BLU.          :rem 126
2790 DATAC'E' QUI UNA CHIAVE ROSSA.,PRIMA
      PUOI DARE UN'OCCHIATA AL LUOGO...
                                  :rem 156
2800 DATASALTA FUORI UN MOSTRO E DICE 'DE
     VI RISPONDERE A QUESTA DOMANDA!
                                  :rem  50
2810 DATA'ORA PRENDERO'TUTTO IL TUO ORO E
      TI PORTERO' VIA!            :rem 198
2820 DATAIL MOSTRO SCOMPARE.
             VEDI DELL'ORO QUI!   :rem  87
2830 DATAC'E' UNA MAPPA QUI!,C'E' UNA CHI
     AVE ROSSA QUI!               :rem  78
2840 DATAC'E' UN FLASH IMPROVVISO E TI TR
     OVI IN MEZZO AD ALCUNI ALBERI.
                                  :rem 173
2850 IFIN(5)=0THENPRINT"NON HAI UNA MAPPA
     !":RETURN                    :rem  51
2860 PRINT"{CLR}":PRINT           :rem 251
2870 FORI=1TO10:TB=1:FORJ=1TO10   :rem 142
2880 IFI=PYANDJ=PXTHENPRINTTAB(TB)"{GRN}*
     ";:GOTO2930                  :rem  20
2890 IFFL(I,J)=3THENPRINTTAB(TB)"{BLK}M";
     :GOTO2930                    :rem 139
2900 IFFL(I,J)=2THENPRINTTAB(TB)"{BLU}R";
     :GOTO2930                    :rem  22
2910 IFFL(I,J)=1THENPRINTTAB(TB)"{PUR}H";
     :GOTO2930                    :rem 137
2920 PRINTTAB(TB)"{RVS}{YEL} {OFF}";
                                  :rem 217
2930 TB=TB+2:NEXTJ:PRINT:PRINT:NEXTI
                                  :rem 167
2940 PRINT"{BLU}":RETURN          :rem 217
```



DAI MEMORIA AL TUO VIC-20



VIXEN SWITCHABLE 16K RAM

A SOLE L. 79.000 + IVA

CON QUESTA CARTRIDGE POTRAI ESPANDERE LA MEMORIA DEL TUO VIC-20 COME VORRAI: 3K-8K-16K

Progettata e prodotta in Inghilterra dalla Stonechip Electronics questa cartuccia, tramite il semplice posizionamento di micro-switch, ti consente un ampliamento differenziato della **RAM: 3K o 8K o 16K,** secondo quanto richiede il programma che devi far girare. I **contatti placcati in oro,** assicurano una lunga durata a questa cartridge che darà nuova vita al tuo VIC-20. Un prezzo introvabile e **garanzia per 6 mesi.**

PROGRAMMERS AID CARTRIDGE

A LIRE 39.500 + IVA

Questa cartuccia aggiunge ben **19 comandi** al tuo **VIC-20:** renumber, autoline, edit, sound, cursor, block, delete .... e altri. E' un prodotto della Stonechip Electronics con **6 mesi di garanzia** integrale.

MODULO DI ESPANSIONE PER VIC-20



PREZZO ECCEZIONALE SOLO L. 62.000 + IVA

Questo modulo di espansione differenziata, prodotto dalla Stonechip Electronics, ti consente di **utilizzare contemporaneamente** (posizionando i microswitch di cui è dotato) da **2 a 4 cartridges.** Aggiunge inoltre alla **ROM 4K.** Inserendo con l'espansione Vixen 16K Ram una espansione standard di 8K si ottengono ben **24K RAM.** E' anche dotato di una presa per collegare un ulteriore modulo di espansione. Garanzia 6 mesi.

TRE CARTRIDGES COMMODORE PER VIC-20 AL PREZZO DI UNA

Scegli i tre videogiochi che preferisci a Lire 41.000 + IVA

Codice 201: Sargon chess - Omega race - Avenger
Codice 202: Omega race - Voodoo castle - Rat race
Codice 203: Ride Fort Knox - Avenger - Mole attack
Codice 204: Gorf - Omega race - Cosmic cruncher

COMPUTER CASSETTE C-20

Nastri magnetici da venti minuti ad altissima affidabilità, specifici per computer. Nastro PE 619 AGFA, scorrimento su perni in acciaio autolubrificanti, assemblate con cinque viti, corpo elegantissimo completamente trasparente, complete di etichette e cartoncino.

12 CASSETTE A L. 11.400 + IVA

INTERFACCIA PER REGISTRATORE PER VIC-20 E PER CBM-64

PREZZO SPECIALE L. 26.000 + IVA



La Stonechip Electronics é stata la prima ditta in Europa a realizzare l'interfaccia per utilizzare qualsiasi registratore con i computer Commodore. Continuamente aggiornata é di certo una delle più affidabili presenti sul mercato. Garanzia di 6 mesi.

LIGHT-PEN PER VIC-20 E CBM-64

La penna ottica é uno dei più collaudati prodotti della Stonechip Electronics. Ha un'ottima stabilità e buona risoluzione verticale/orizzontale. Completa di cassetta programmi. Un prezzo conveniente e una garanzia di 6 mesi.

LIRE 54.000 + IVA

50 FANTASTICI VIDEOGIOCHI 50 PER VIC-20 O CBM-64

Registrati su nastro, della software-house inglese Cascade Games, all'**inverosimile prezzo** di sole L. **25.000 + IVA.**

Spedire il presente **MODULO D'ORDINE,** o fotocopia, in una busta chiusa, unendo L. 2.600 in francobolli per spese postali.
UN OMAGGIO A CHI ORDINA ALMENO DUE ARTICOLI

Spett.le **APCO s.r.l. - Cas. Post. 239 - 10015 IVREA (TO)**
desidero ricevere quanto da me contrassegnato con X. Pagheró direttamente al postino gli importi qui elencati, che sono comprensivi di IVA e di spese di imballo e contrassegno. sc

☐ Vixen Switchable 16K RAM	a L. 95.000
☐ Programmer Aid Cartridge	a L. 48.000
☐ Modulo di espansione per VIC-20	a L. 75.000
☐ 3 cartridges Commodore codice	a L. 50.000
☐ 12 computer cassette C-20	a L. 15.000
☐ Interfaccia per registratore	a L. 31.000
☐ Light-Pen per VIC-20 e CBM-64	a L. 65.000
☐ 50 videogiochi per ☐ VIC-20 ☐ CBM-64	a L. 30.000

☐ Non acquisto nulla ma desidero ricevere i Vostri cataloghi

Nome e Cognome ____________________

Via ____________________

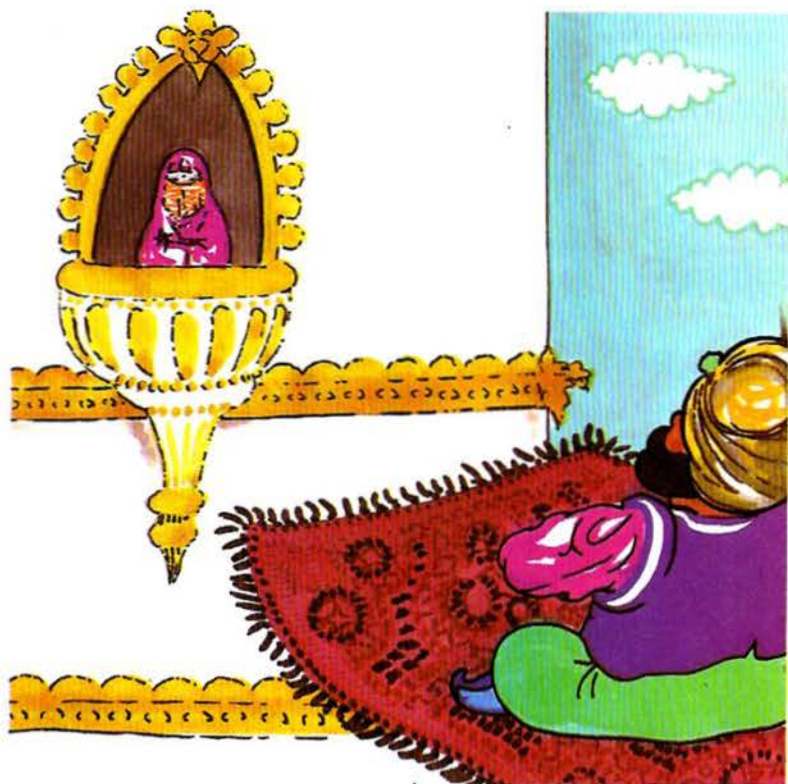
Città ____________________ CAP. ________

data ____________ firma ____________________

Rescue of Blondell

di **G. Albrecht**
trad. ed adatt.
di **M. Cristuib Grizzi**

Ecco un gioco dall'azione velocissima e dotato di scorrimento orizzontale dello schermo pixel per pixel. Le due versioni per VIC 20 (con almeno 8 Kbyte di espansione di memoria) e per C64 sono costituite da due caricatori BASIC di linguaggio macchina e richiedono l'uso del joystick (porta 2 per il C64).



La figlia del Califfo di Bagdad, Blondell, è stata rapita e tocca a voi, prodi difensori dell'ordine, andare in sua ricerca e procedere al suo salvataggio. La principessa è imprigionata in una torre, ed il vostro compito consiste nel volare fino ad essa sul vostro tappeto magico, caricare la principessa e tornare indietro con lei fino alla base. Non sarà un'impresa facile, poiché il rapitore – un noto genio del male – ha scatenato stormi di leggeri, ma potentissimi uccellacci pronti a combattervi. Questi uccellacci sono intelligenti e cercheranno di salire sul vostro tappeto volante, oltre a lanciare verso di voi lampi di energia. Tutttavia anche voi siete protetti da una buone dose di magia, che vi rende immuni dagli attacchi, ma attenzione! Ogni attacco riuscito da parte degli uccellacci provocherà l'indebolimento del vostro invisibile scudo protettivo, fino a che potrete contare soltanto su voi stessi.

Siete comunque in grado di lanciare a vostra volta lampi di energia all'indirizzo degli uccellacci, cercando di abbatterli.

"Rescue of Blondell" è un gioco con caratteristiche di scorrimento orizzontale dello schermo pixel per pixel e possibilità di usufruire di diversi livelli di difficoltà, proporzionali al numero degli uccellacci (da 1 a 3 nella versione per C64, e da 1 a 9 nella versione per VIC 20).

Una volta iniziato il gioco volerete verso la parte destra dello schermo, spingendo il joystick in quella direzione; nella parte alta dello schermo sono visualizzati il punteggio, la quantità di magia che ancora vi protegge ed i punti di bonus che ricevete portando in salvo Blondell. Guadagnate 10 punti per ogni uccellaccio abbattuto, ed il punteggio di bonus diminuisce man mano che passa il tempo, costringendovi ad agire con rapidità.

Ricordate che il Califfo vi ha ordinato di ritornare con la principessa, così, se cercate di rientrare alla base senza di lei, potreste avere una spiacevole sorpresa.

Come detto, la versione per VIC 20 richiede l'espansione di memoria da almeno 8 Kbyte e richiede di riconfigurare la memoria prima del caricamento del programma. Seguite attentamente questa procedura:

1. Spegnete il VIC 20
2. Inserite l'espansione di memoria da almeno 8 Kbyte.

3. Digitate in modo diretto le seguenti istruzioni:
POKE44,32: POKE32*256,0:NEW [return]
POKE648,30:SYS58648 [return]
4. Caricate il programma e date il RUN.

Nella versione per VIC 20 gli uccellacci non sparano lampi di energia, ma possono essere in numero tale da crearvi seri problemi nel completamento della vostra missione.
Nella versione per C64 la principessa si trova in cima ad una torre: per salvarla avvicinatevi semplicemente a lei. Se avete agito con la necessaria delicatezza, Blondell scompare e voi diventate di colore blu. Nella versione per VIC 20, invece, Blondell è imprigionata in fondo ad un profondo tunnel: per salvarla dovete volare fino in fondo ad esso ed atterrare; si aprirà una porta segreta e la principessa diventerà visibile. Toccatela semplicemente per salvarla e per caricarla sul tappeto e portarla sana e salva alla base (uccellacci permettendo...)

Rescue of Blondell versione per VIC 20

```
10 PRINT"{CLR}{ 3 GIU'}RESCUE OF BLONDELL
   ":PRINT"{ 2 GIU'}ATTENDERE..."
                                :rem 243
20 FORI=4109TO5812              :rem  16
30 READA:CK=CK+A:POKEI,A:NEXT   :rem  87
40 IFCK<>176773THENPRINT"ERRORE NELLE IST
   RUZIONI DATA!!!":END         :rem  39
50 SYS4109                      :rem  50
4109 DATA 32,12,19,32,58,18,32,127
                                :rem 136
4117 DATA 16,32,150,18,32,68,19,32
                                :rem 136
4125 DATA 222,20,32,181,16,32,117,19
                                :rem 224
4133 DATA 32,3,19,32,117,19,32,19
                                :rem  82
4141 DATA 20,32,128,21,173,141,2,208
                                :rem 218
4149 DATA 251,165,4,240,229,165,2,201
                                :rem  28
4157 DATA 10,208,223,165,0,201,229,208
                                :rem  71
4165 DATA 217,32,170,16,160,21,185,12
                                :rem  23
4173 DATA 22,153,132,30,136,16,247,165
                                :rem  77
4181 DATA 65,24,101,63,133,65,165,66
                                :rem 242
4189 DATA 101,64,133,66,32,68,19,160
                                :rem 247
4197 DATA 10,185,56,22,153,181,30,136
                                :rem  33
4205 DATA 16,247,32,128,21,165,203,201
                                :rem  68
4213 DATA 11,240,149,201,28,208,246,76
                                :rem  78
4221 DATA 34,253,169,147,32,210,255,32
                                :rem  77
4229 DATA 68,19,160,18,185,67,22,153
                                :rem 255
4237 DATA 222,30,136,16,247,32,249,253
                                :rem  83
4245 DATA 164,203,185,94,236,201,49,144
                                :rem 140
4253 DATA 247,201,58,176,243,56,233,48
                                :rem  95
4261 DATA 10,141,72,3,96,169,32,160
                                :rem 185
4269 DATA 0,153,0,30,136,208,250,96
                                :rem 185
4277 DATA 166,1,164,2,32,51,17,32
                                :rem  85
4285 DATA 95,17,169,0,141,19,145,173
                                :rem 252
4293 DATA 17,145,74,74,74,176,5,192
                                :rem 210
4301 DATA 2,144,1,136,74,72,176,19
                                :rem 136
4309 DATA 200,192,19,208,14,136,165,3
                                :rem  30
4317 DATA 208,9,169,1,133,3,169,36
                                :rem 148
4325 DATA 141,154,31,104,74,72,176,15
                                :rem  29
4333 DATA 169,33,133,109,224,7,176,6
                                :rem 245
4341 DATA 32,245,17,76,252,16,202,104
                                :rem  24
4349 DATA 74,176,4,169,1,133,113,169
                                :rem 252
4357 DATA 127,141,34,145,44,32,145,48
                                :rem  37
4365 DATA 15,169,34,133,109,224,15,144
                                :rem  85
4373 DATA 6,32,176,17,76,29,17,232
                                :rem 151
4381 DATA 134,1,132,2,32,51,17,32
                                :rem  71
4389 DATA 95,17,165,109,160,0,145,251
                                :rem  43
4397 DATA 169,2,32,82,19,96,32,79
                                :rem 116
4405 DATA 17,32,104,18,24,138,101,251
                                :rem  18
4413 DATA 133,251,165,252,105,0,133,252
                                :rem 116
4421 DATA 32,122,18,136,208,250,32,87
                                :rem  27
4429 DATA 17,96,120,133,105,134,106,132
                                :rem 125
4437 DATA 107,96,165,105,166,106,164,107
                                :rem 192
4445 DATA 88,96,32,79,17,160,0,177
                                :rem 161
4453 DATA 251,240,20,201,38,240,8,201
                                :rem  13
4461 DATA 36,208,56,169,1,133,4,198
                                :rem 201
4469 DATA 69,165,69,201,0,208,44,169
                                :rem   4
4477 DATA 15,141,14,144,169,39,145,251
                                :rem  92
4485 DATA 32,82,19,88,165,162,105,100
                                :rem  40
4493 DATA 197,162,208,252,32,170,16,169
                                :rem 147
4501 DATA 0,160,21,133,63,133,64,185
                                :rem 225
4509 DATA 34,22,153,132,30,136,16,247
                                :rem  27
4517 DATA 76,97,16,169,32,145,251,32
                                :rem 255
4525 DATA 87,17,96,32,79,17,32,104
                                :rem 155
4533 DATA 18,165,0,201,10,240,56,162
                                :rem 224
4541 DATA 19,160,1,177,251,208,18,169
                                :rem  40
4549 DATA 32,145,251,136,208,3,32,98
                                :rem 248
4557 DATA 19,169,0,145,251,32,82,19
                                :rem 200
```

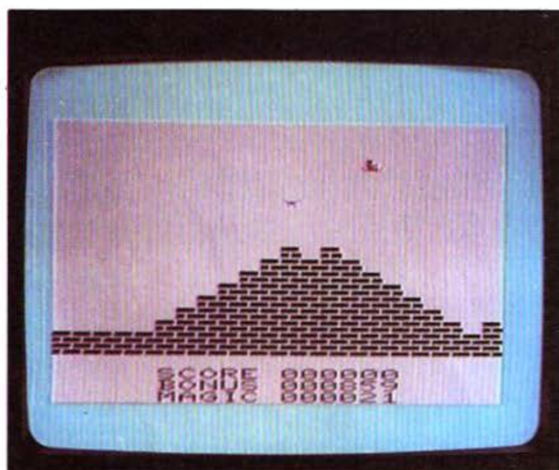
LOAD 

Rescue of Blondell

```
4565 DATA 200,200,192,22,208,229,189,180
                                    :rem 184
4573 DATA 23,136,145,251,169,0,32,82
                                    :rem 241
4581 DATA 19,32,122,18,202,208,210,198
                                    :rem  80
4589 DATA 0,32,58,18,32,87,17,96:rem  62
4597 DATA 32,79,17,32,104,18,165,0
                                    :rem 147
4605 DATA 201,229,240,56,162,19,160,20
                                    :rem  74
4613 DATA 177,251,208,20,169,32,145,251
                                    :rem 134
4621 DATA 200,192,21,208,3,32,98,19
                                    :rem 185
4629 DATA 169,0,145,251,32,82,19,136
                                    :rem 248
4637 DATA 136,16,229,189,200,23,200,145
                                    :rem 133
4645 DATA 251,169,0,32,82,19,32,122
                                    :rem 188
4653 DATA 18,202,208,210,230,0,32,58
                                    :rem 225
4661 DATA 18,32,87,17,96,169,32,160
                                    :rem 207
4669 DATA 19,153,180,23,153,200,23,136
                                    :rem  82
4677 DATA 16,247,164,0,192,17,240,12
                                    :rem 242
4685 DATA 185,181,22,168,169,0,153,180
                                    :rem  98
4693 DATA 23,136,16,250,164,0,185,202
                                    :rem  30
4701 DATA 22,168,169,0,153,200,23,136
                                    :rem  23
4709 DATA 16,250,96,169,0,133,251,169
                                    :rem  46
4717 DATA 30,133,252,169,0,133,110,173
                                    :rem  72
4725 DATA 3,144,16,251,96,165,251,24
                                    :rem 244
4733 DATA 105,22,133,251,165,252,105,0
                                    :rem  67
4741 DATA 133,252,96,165,251,56,233,22
                                    :rem  88
4749 DATA 133,251,165,252,233,0,133,252
                                    :rem 130
4757 DATA 96,169,10,133,2,169,7,133
                                    :rem 204
4765 DATA 1,169,147,32,210,255,169,0
                                    :rem 245
4773 DATA 133,113,133,114,133,3,133,4
                                    :rem  17
4781 DATA 168,153,220,23,136,208,250,169
                                    :rem 190
4789 DATA 34,133,109,133,116,160,22,32
                                    :rem  81
4797 DATA 176,17,136,16,250,160,0,185
                                    :rem  44
4805 DATA 0,128,153,0,28,185,0,129
                                    :rem 135
4813 DATA 153,0,29,136,208,241,169,255
                                    :rem  89
4821 DATA 141,5,144,160,79,185,101,22
                                    :rem  28
4829 DATA 153,8,29,136,16,247,160,7
                                    :rem 205
4837 DATA 185,173,22,153,0,28,136,16
                                    :rem 245
4845 DATA 247,169,232,133,63,169,3,133
                                    :rem  97
4853 DATA 64,169,0,133,65,133,66,133
                                    :rem 248
4861 DATA 70,169,25,133,69,96,165,162
                                    :rem  57
4869 DATA 105,5,197,162,208,252,96,160
                                    :rem 103
4877 DATA 12,169,8,153,167,23,136,16
                                    :rem   0
4885 DATA 250,160,242,162,9,138,153,181
                                    :rem 142
4893 DATA 22,32,58,19,201,150,176,8
                                    :rem 197
4901 DATA 224,3,144,9,202,76,50,19
                                    :rem 137
4909 DATA 224,16,176,1,232,136,208,229
                                    :rem  90
4917 DATA 169,252,133,0,96,165,108,10
                                    :rem  40
4925 DATA 10,56,101,108,133,108,96,169
                                    :rem  88
4933 DATA 0,160,0,153,0,150,153,0
                                    :rem  64
4941 DATA 151,136,208,247,96,72,165,251
                                    :rem 148
4949 DATA 133,43,165,252,24,105,120,133
                                    :rem 129
4957 DATA 44,104,145,43,96,165,110,208
                                    :rem  92
4965 DATA 14,32,136,18,169,32,145,251
                                    :rem  42
4973 DATA 32,122,18,169,1,133,110,96
                                    :rem 241
4981 DATA 165,114,208,44,165,113,240,39
                                    :rem 139
4989 DATA 165,251,133,253,165,252,133,254
                                    :rem 248
4997 DATA 169,1,133,115,169,15,141,14
                                    :rem  43
5005 DATA 144,165,109,201,33,208,6,169
                                    :rem  80
5013 DATA 1,133,111,208,4,169,0,133
                                    :rem 167
5021 DATA 111,169,7,133,112,133,114,96
                                    :rem  71
5029 DATA 177,253,240,99,165,115,208,6
                                    :rem  98
5037 DATA 160,0,169,32,145,253,169,0
                                    :rem 237
5045 DATA 133,115,165,111,208,9,230,253
                                    :rem 123
5053 DATA 208,2,230,254,76,206,19,198
                                    :rem  40
5061 DATA 253,165,253,201,255,208,2,198
                                    :rem 135
5069 DATA 254,165,253,133,251,165,254,133
                                    :rem 240
5077 DATA 252,198,112,165,112,240,48,10
                                    :rem 131
5085 DATA 10,10,10,24,105,158,141,13
                                    :rem 215
5093 DATA 144,177,253,240,34,201,38,208
                                    :rem 134
```

```
5101 DATA 20,32,114,21,169,39,145,253 :rem 21
5109 DATA 169,7,32,82,19,169,15,141 :rem 201
5117 DATA 14,144,76,12,20,169,35,145 :rem 238
5125 DATA 253,169,0,32,82,19,96,169 :rem 205
5133 DATA 0,133,113,133,114,96,160,0 :rem 217
5141 DATA 132,25,185,220,23,170,133,20 :rem 62
5149 DATA 185,221,23,168,133,21,132,26 :rem 80
5157 DATA 32,51,17,160,0,177,251,76 :rem 190
5165 DATA 120,20,169,32,145,251,164,26 :rem 78
5173 DATA 32,58,19,201,215,144,114,228 :rem 80
5181 DATA 1,240,7,144,4,202,76,71 :rem 81
5189 DATA 20,232,196,2,240,7,144,4 :rem 138
5197 DATA 136,76,82,20,200,132,26,32 :rem 236
5205 DATA 51,17,32,148,20,169,38,160 :rem 238
5213 DATA 0,145,251,169,6,32,82,19 :rem 138
5221 DATA 164,25,138,153,220,23,165,26 :rem 76
5229 DATA 153,221,23,200,200,204,72,3 :rem 9
5237 DATA 208,158,96,201,38,240,179,165 :rem 151
5245 DATA 21,240,175,164,25,32,58,19 :rem 242
5253 DATA 74,74,74,74,153,220,23,169 :rem 251
5261 DATA 0,153,221,23,76,112,20,160 :rem 217
5269 DATA 0,177,251,201,32,240,17,201 :rem 20
5277 DATA 33,240,13,201,34,240,9,166 :rem 233
5285 DATA 20,164,21,132,26,32,51,17 :rem 179
5293 DATA 96,32,58,19,201,60,176,8 :rem 155
5301 DATA 224,20,240,4,232,76,219,20 :rem 221
5309 DATA 201,120,176,8,224,0,240,4 :rem 173
5317 DATA 202,76,219,20,201,180,176,5 :rem 28
5325 DATA 192,1,240,1,136,201,181,144 :rem 17
5333 DATA 5,192,20,240,1,200,76,82 :rem 127
5341 DATA 20,120,169,235,141,20,3,169 :rem 21
5349 DATA 20,141,21,3,88,96,165,251 :rem 196
5357 DATA 72,165,252,72,198,116,208,3 :rem 48
5365 DATA 32,38,21,160,7,165,116,201 :rem 233
5373 DATA 10,144,20,173,14,144,240,3 :rem 222
5381 DATA 206,14,144,185,141,22,153,48 :rem 82
5389 DATA 29,136,16,247,76,29,21,185 :rem 9
5397 DATA 133,22,153,48,29,136,16,247 :rem 47
5405 DATA 104,133,252,104,133,251,76,191 :rem 173
5413 DATA 234,169,20,133,116,32,104,18 :rem 72
5421 DATA 162,19,160,21,177,251,201,39 :rem 77
5429 DATA 144,27,201,42,176,23,24,105 :rem 28
5437 DATA 1,72,169,130,141,13,144,104 :rem 23
5445 DATA 201,42,208,2,169,32,145,251 :rem 28
5453 DATA 169,7,32,82,19,136,16,220 :rem 196
5461 DATA 32,122,18,202,16,212,198,63 :rem 28
5469 DATA 165,63,201,255,208,14,198,64 :rem 101
5477 DATA 165,64,201,255,208,6,169,0 :rem 250
5485 DATA 133,63,133,64,96,165,65,24 :rem 1
5493 DATA 105,10,133,65,165,66,105,0 :rem 234
5501 DATA 133,66,96,165,65,133,73,165 :rem 47
5509 DATA 66,133,74,169,14,133,77,32 :rem 254
5517 DATA 197,21,165,63,133,73,165,64 :rem 48
5525 DATA 133,74,169,36,133,77,32,197 :rem 53
5533 DATA 21,165,69,133,73,165,70,133 :rem 39
5541 DATA 74,169,58,133,77,32,197,21 :rem 3
5549 DATA 160,4,185,86,22,153,189,31 :rem 255
5557 DATA 185,91,22,153,211,31,185,96 :rem 46
5565 DATA 22,153,233,31,136,16,235,96 :rem 38
5573 DATA 169,0,133,98,133,99,133,100 :rem 43
5581 DATA 162,15,6,73,38,74,120,248 :rem 199
5589 DATA 165,98,101,98,133,98,165,99 :rem 79
5597 DATA 101,99,133,99,165,100,101,100 :rem 133
5605 DATA 133,100,216,88,202,16,227,162 :rem 126
5613 DATA 2,181,98,72,74,74,74,74 :rem 111
5621 DATA 32,2,22,104,41,15,32,2:rem 14
5629 DATA 22,202,16,237,96,230,77,164 :rem 44
5637 DATA 77,9,48,153,180,31,96,25 :rem 162
5645 DATA 15,21,32,19,1,22,5,4 :rem 184
5653 DATA 32,20,8,5,32,16,18,9 :rem 195
5661 DATA 14,3,5,19,19,16,15,15 :rem 244
5669 DATA 18,32,16,18,9,14,3,5 :rem 204
5677 DATA 19,19,32,2,12,15,14,4 :rem 244
5685 DATA 5,12,12,16,12,1,25,32 :rem 232
5693 DATA 1,7,1,9,14,63,14,21 :rem 141
5701 DATA 13,2,5,18,32,15,6,32 :rem 181
5709 DATA 2,9,18,4,19,32,49,45 :rem 208
```

```
5717 DATA 57,19,3,15,18,5,2,15   :rem  199
5725 DATA 14,21,19,13,1,7,9,3    :rem  139
5733 DATA 12,12,4,60,12,189,126,0
                                 :rem   78
5741 DATA 48,48,32,60,48,189,126,0
                                 :rem  153
5749 DATA 0,0,84,170,0,0,0,0     :rem   75
5757 DATA 24,36,90,60,90,24,60,126
                                 :rem  146
5765 DATA 0,0,195,60,24,36,0,0   :rem  185
5773 DATA 0,0,0,126,153,36,0,0   :rem  175
5781 DATA 0,0,68,60,94,56,64,0   :rem  201
5789 DATA 0,38,124,60,124,58,72,0
                                 :rem   96
5797 DATA 194,102,252,46,255,90,44,69
                                 :rem   58
5805 DATA 254,254,254,0,239,239,239,0
                                 :rem   43
```



Rescue of Blondell versione per C64

```
10 PRINT"{CLR}{ 3 GIU'}"TAB(11)"[<5>]
   {RVS}RESCUE OF BLONDELL"      :rem   24
20 PRINT"{ 4 GIU'}"TAB(11)"{CYN}{RVS}STO
   INSERENDO IL L/M..."          :rem  122
30 PRINT"{ 4 GIU'}"TAB(11)"{CYN}{RVS}ATTE
   NDERE PREGO..."               :rem  154
40 FORI=49152TO51972             :rem  129
50 READA:CK=CK+A:POKEI,A:NEXT    :rem   89
60 IFCK<>318395THENPRINT"{ 3 GIU'}ERRORE
   NELLE ISTRUZIONI DATA!!!":STOP
                                 :rem  201
70 SYS49152                      :rem  107
49152 DATA 76,46,202,32,145,196,32
                                 :rem  150
49159 DATA 151,195,32,142,194,32,197
                                 :rem    4
49166 DATA 193,32,222,193,32,164,195
                                 :rem    0
49173 DATA 32,209,201,173,31,208,165
                                 :rem  242
49180 DATA 2,208,57,173,60,3,201:rem   39
49187 DATA 232,240,3,76,125,192,173
                                 :rem  203
49194 DATA 61,3,201,65,240,3,76 :rem  251
49201 DATA 125,192,162,50,32,50,193
                                 :rem  187
49208 DATA 202,208,250,169,6,141,39
                                 :rem  198
49215 DATA 208,162,32,142,2,68,142
                                 :rem  141
49222 DATA 3,68,142,2,70,142,3  :rem  192
49229 DATA 70,169,1,133,2,32,164:rem   45
49236 DATA 195,76,125,192,169,13,205
                                 :rem    5
49243 DATA 60,3,169,64,237,61,3 :rem    1
49250 DATA 144,25,173,1,208,201,118
                                 :rem  185
49257 DATA 144,18,173,0,208,201,114
                                 :rem  189
49264 DATA 176,11,169,168,133,113,169
                                 :rem   54
49271 DATA 198,133,114,76,107,202,32
                                 :rem  246
49278 DATA 60,201,32,174,200,32,78
                                 :rem  144
49285 DATA 199,32,53,196,206,248,207
                                 :rem   13
49292 DATA 208,8,32,35,200,169,2:rem   47
49299 DATA 141,248,207,32,60,201,32
                                 :rem  195
49306 DATA 174,200,162,0,189,249,7
                                 :rem  151
49313 DATA 201,250,208,40,222,170,2
                                 :rem  174
49320 DATA 208,35,169,253,157,52,3
                                 :rem  151
49327 DATA 169,252,157,55,3,138,10
                                 :rem  155
49334 DATA 168,169,25,153,3,208,173
                                 :rem  208
49341 DATA 27,212,174,60,3,224,15
                                 :rem   88
49348 DATA 176,4,201,120,144,242,153
                                 :rem  240
49355 DATA 2,208,232,228,20,208,204
                                 :rem  189
49362 DATA 173,31,208,72,74,144,20
                                 :rem  146
49369 DATA 32,211,202,169,10,133,113
                                 :rem  235
49376 DATA 169,199,133,114,169,0,133
                                 :rem    6
49383 DATA 106,133,107,76,107,202,104
                                 :rem   33
49390 DATA 13,30,208,74,144,13,206
                                 :rem  140
49397 DATA 167,2,174,167,2,224,255
                                 :rem  162
49404 DATA 208,3,76,217,192,160,0
                                 :rem   94
49411 DATA 74,72,144,26,185,249,7
                                 :rem  110
49418 DATA 201,250,240,19,32,50,193
                                 :rem  189
49425 DATA 169,20,153,170,2,169,250
                                 :rem  198
49432 DATA 153,52,3,153,55,3,153:rem   42
49439 DATA 249,7,104,200,196,20,208
                                 :rem  200
49446 DATA 220,173,141,2,208,251,32
                                 :rem  187
49453 DATA 68,193,76,27,192,173,168
                                 :rem  226
49460 DATA 2,24,105,10,141,168,2:rem   31
49467 DATA 173,169,2,105,0,141,169
                                 :rem  153
49474 DATA 2,96,173,168,2,133,253
                                 :rem  108
```

```
49481 DATA 173,169,2,133,254,169,130 :rem 255
49488 DATA 141,249,207,32,120,193,173 :rem 49
49495 DATA 167,2,133,253,169,0,133 :rem 153
49502 DATA 254,169,147,141,249,207,32 :rem 48
49509 DATA 120,193,165,106,133,253,165 :rem 93
49516 DATA 107,133,254,169,218,141,249 :rem 101
49523 DATA 207,32,120,193,96,160,9 :rem 150
49530 DATA 169,0,141,32,203,165,253 :rem 189
49537 DATA 217,174,193,165,254,249,175 :rem 116
49544 DATA 193,144,20,165,253,56,249 :rem 4
49551 DATA 174,193,133,253,165,254,249 :rem 106
49558 DATA 175,193,133,254,238,32,203 :rem 52
49565 DATA 208,224,173,32,203,32,185 :rem 249
49572 DATA 193,169,0,141,32,203,136 :rem 196
49579 DATA 136,16,209,96,1,0,10 :rem 1
49586 DATA 0,100,0,232,3,16,39 :rem 192
49593 DATA 238,249,207,174,249,207,9 :rem 18
49600 DATA 48,157,208,6,96,160,0:rem 52
49607 DATA 169,0,133,251,169,64,133 :rem 204
49614 DATA 252,162,40,169,32,145,251 :rem 247
49621 DATA 200,208,249,230,252,202,208 :rem 79
49628 DATA 244,96,160,0,169,0,133 :rem 102
49635 DATA 251,169,64,133,252,169,0 :rem 207
49642 DATA 145,251,32,121,194,200,192 :rem 34
49649 DATA 16,208,246,169,0,133,251 :rem 206
49656 DATA 169,86,133,252,145,251,32 :rem 5
49663 DATA 121,194,200,192,27,208,246 :rem 45
49670 DATA 169,0,145,251,32,121,194 :rem 197
49677 DATA 200,208,66,230,252,232,224 :rem 40
49684 DATA 2,208,59,169,1,141,2 :rem 4
49691 DATA 68,169,2,141,3,68,169:rem 70
49698 DATA 3,141,2,70,169,4,141 :rem 2
49705 DATA 3,70,169,32,141,0,68 :rem 254
49712 DATA 141,1,68,141,0,70,141:rem 33
49719 DATA 0,66,141,1,66,141,2 :rem 199
49726 DATA 66,141,1,70,141,3,66 :rem 253
49733 DATA 160,16,169,5,153,0,86:rem 54
49740 DATA 200,192,22,208,248,96,173 :rem 254
49747 DATA 27,212,201,85,144,173,201 :rem 246
49754 DATA 160,176,14,165,252,201,72 :rem 250
49761 DATA 144,163,56,233,2,133,252 :rem 197
49768 DATA 76,6,194,165,252,201,96 :rem 172
```

LOAD

Rescue of Blondell

```
49775 DATA 176,149,24,105,2,133,252 :rem 204
49782 DATA 76,6,194,165,252,72,169 :rem 175
49789 DATA 0,145,251,230,252,230,252 :rem 245
49796 DATA 165,252,201,136,144,242,104 :rem 93
49803 DATA 133,252,96,169,4,141,60 :rem 153
49810 DATA 3,169,64,141,61,3,169:rem 53
49817 DATA 147,32,210,255,160,255,169 :rem 50
49824 DATA 14,153,255,215,153,254,216 :rem 44
49831 DATA 153,253,217,153,252,218,136 :rem 95
49838 DATA 208,241,169,1,141,33,208 :rem 202
49845 DATA 162,24,169,0,157,0,212 :rem 97
49852 DATA 202,16,250,169,129,141,18 :rem 251
49859 DATA 212,169,200,141,15,212,169 :rem 45
49866 DATA 15,141,24,212,160,4,162 :rem 142
49873 DATA 21,24,32,240,255,160,198 :rem 202
49880 DATA 169,152,32,30,171,160,20 :rem 193
49887 DATA 162,21,24,32,240,255,160 :rem 198
49894 DATA 198,169,160,32,30,171,160 :rem 5
49901 DATA 12,162,23,24,32,240,255 :rem 135
49908 DATA 160,199,169,70,32,30,171 :rem 210
49915 DATA 169,129,141,26,208,169,127 :rem 60
49922 DATA 141,13,220,169,0,141,40 :rem 132
49929 DATA 208,141,41,208,141,42,208 :rem 248
49936 DATA 141,45,208,141,44,208,141 :rem 248
49943 DATA 43,208,169,11,141,46,208 :rem 204
49950 DATA 169,251,141,255,7,141,254 :rem 255
49957 DATA 7,141,253,7,141,252,7:rem 56
49964 DATA 169,33,141,4,212,169,208 :rem 209
49971 DATA 133,106,169,7,133,107,169 :rem 4
49978 DATA 0,133,108,169,16,141,5 :rem 105
49985 DATA 212,169,240,141,6,212,169 :rem 1
```

```
49992 DATA 15,141,24,212,162,2,169
                              :rem 149
49999 DATA 253,157,52,3,169,252,157
                              :rem 225
50006 DATA 55,3,202,16,243,169,0:rem 30
50013 DATA 141,18,208,173,17,208,41
                              :rem 182
50020 DATA 127,141,17,208,160,7,169
                              :rem 185
50027 DATA 0,153,0,57,136,16,250:rem 29
50034 DATA 160,47,185,104,198,153,0
                              :rem 190
50041 DATA 56,136,16,247,169,2,141
                              :rem 141
50048 DATA 35,208,169,0,141,34,208
                              :rem 140
50055 DATA 133,2,169,200,141,167,2
                              :rem 130
50062 DATA 169,0,141,168,2,141,169
                              :rem 140
50069 DATA 2,96,120,169,123,141,20
                              :rem 136
50076 DATA 3,169,199,141,21,3,88:rem 54
50083 DATA 96,173,60,3,141,201,195
                              :rem 141
50090 DATA 173,61,3,141,202,195,169
                              :rem 190
50097 DATA 0,141,204,195,169,4,141
                              :rem 140
50104 DATA 205,195,162,19,169,0,141
                              :rem 187
50111 DATA 63,3,173,17,208,16,251
                              :rem 82
50118 DATA 160,39,185,0,0,153,0 :rem 235
50125 DATA 0,136,16,247,173,204,195
                              :rem 189
50132 DATA 24,105,40,141,204,195,173
                              :rem 227
50139 DATA 205,195,105,0,141,205,195
                              :rem 236
50146 DATA 238,202,195,238,202,195,202
                              :rem 86
50153 DATA 208,219,96,206,62,3,16
                              :rem 95
50160 DATA 50,238,60,3,208,3,238:rem 35
50167 DATA 61,3,169,7,141,62,3  :rem 199
50174 DATA 32,164,195,76,35,196,238
                              :rem 212
50181 DATA 62,3,173,62,3,201,8  :rem 189
50188 DATA 208,21,206,60,3,173,60
                              :rem 87
50195 DATA 3,201,255,208,3,206,61
                              :rem 84
50202 DATA 3,169,0,141,62,3,32  :rem 180
50209 DATA 164,195,32,46,196,96,133
                              :rem 209
50216 DATA 38,134,39,132,40,96,165
                              :rem 148
50223 DATA 38,166,39,164,40,96,174
                              :rem 156
50230 DATA 0,208,172,1,208,169,0:rem 27
50237 DATA 133,251,173,0,220,74,176
                              :rem 186
50244 DATA 5,192,47,144,1,136,74:rem 44
50251 DATA 176,5,192,175,176,1,200
                              :rem 140
50258 DATA 74,176,21,72,169,255,141
                              :rem 205
50265 DATA 248,7,104,224,107,208,9
                              :rem 145
50272 DATA 32,39,196,32,4,196,76:rem 57
50279 DATA 106,196,202,74,176,21,72
                              :rem 201
50286 DATA 169,254,141,248,7,104,224
                              :rem 250
50293 DATA 205,208,9,32,39,196,32
                              :rem 101
50300 DATA 236,195,76,130,196,232,74
                              :rem 247
50307 DATA 176,5,169,1,141,128,3:rem 41
50314 DATA 142,0,208,140,1,208,96
                              :rem 79
50321 DATA 160,255,185,231,196,153,127
                              :rem 87
50328 DATA 62,185,104,197,153,0,63
                              :rem 146
50335 DATA 136,208,241,169,255,141,21
                              :rem 34
50342 DATA 208,169,2,141,39,208,169
                              :rem 200
50349 DATA 254,141,248,7,169,118,141
                              :rem 253
50356 DATA 1,208,169,138,141,0,208
                              :rem 140
50363 DATA 173,22,208,41,247,141,22
                              :rem 184
50370 DATA 208,169,4,141,62,3,169
                              :rem 96
50377 DATA 0,141,63,3,169,20,141:rem 34
50384 DATA 3,208,141,5,208,141,7:rem 37
50391 DATA 208,169,80,141,2,208,169
                              :rem 200
50398 DATA 160,141,4,208,169,240,141
                              :rem 242
50405 DATA 6,208,96,144,0,2,64  :rem 196
50412 DATA 0,16,184,0,38,20,0   :rem 127
50419 DATA 0,41,16,36,4,68,80   :rem 148
50426 DATA 2,130,36,1,64,168,0  :rem 188
50433 DATA 176,0,1,52,64,0,240  :rem 183
50440 DATA 16,95,104,208,196,0,128
                              :rem 142
50447 DATA 73,128,0,166,34,160,141
                              :rem 140
50454 DATA 72,32,1,34,74,0,202  :rem 185
50461 DATA 41,0,0,160,0,0,2     :rem 16
50468 DATA 0,0,18,0,0,0,0       :rem 178
50475 DATA 0,0,0,0,0,0,0        :rem 119
50482 DATA 0,0,0,0,0,0,0        :rem 117
50489 DATA 0,0,0,0,0,0,0        :rem 124
50496 DATA 0,0,0,2,34,0,5       :rem 184
50503 DATA 85,64,0,136,128,0,0  :rem 187
50510 DATA 0,0,0,0,0,0,0        :rem 109
50517 DATA 0,0,0,0,0,0,0        :rem 116
50524 DATA 0,0,0,0,0,0,0        :rem 114
50531 DATA 0,0,0,0,0,0,0        :rem 112
50538 DATA 0,0,0,0,0,60,0       :rem 173
50545 DATA 0,66,0,0,66,0,0      :rem 237
50552 DATA 90,0,3,195,192,12,66 :rem 249
50559 DATA 48,16,66,8,32,66,4   :rem 166
50566 DATA 73,165,146,146,36,73,164
                              :rem 210
50573 DATA 24,37,168,189,21,168,195
                              :rem 216
50580 DATA 21,81,0,138,2,0,64   :rem 136
50587 DATA 2,0,64,5,0,160,0     :rem 35
50594 DATA 0,0,0,0,0,255,0      :rem 229
50601 DATA 0,0,0,0,0,0,60       :rem 164
50608 DATA 0,0,66,0,128,66,1    :rem 89
50615 DATA 126,90,126,33,195,132,192
                              :rem 245
50622 DATA 66,3,32,66,4,32,66   :rem 150
50629 DATA 4,80,36,10,24,36,24  :rem 195
50636 DATA 20,24,40,3,189,192,0 :rem 243
50643 DATA 195,0,1,0,128,2,0    :rem 81
50650 DATA 64,2,0,64,5,0,160    :rem 84
```

```
50657 DATA 0,0,0,0,0,0,255          :rem 229
50664 DATA 0,0,0,0,0,0,0            :rem 119
50671 DATA 0,0,15,128,0,31,192      :rem 182
50678 DATA 0,31,192,0,31,224,0      :rem 184
50685 DATA 15,240,0,3,224,96,3      :rem 195
50692 DATA 224,127,3,255,195,3,255  :rem 151
50699 DATA 252,3,224,0,3,224,0      :rem 190
50706 DATA 3,224,0,59,255,156,67:rem 50
50713 DATA 255,130,131,255,129,71,24 :rem 237
50720 DATA 226,56,231,28,0,0,0      :rem 181
50727 DATA 184,0,0,0,0,0,0          :rem 228
50734 DATA 0,0,0,0,1,240,0          :rem 220
50741 DATA 3,248,0,3,248,0,7        :rem 92
50748 DATA 248,0,15,240,6,7,192 :rem 253
50755 DATA 254,7,192,195,255,192,63 :rem 214
50762 DATA 255,192,0,7,192,0,7      :rem 200
50769 DATA 192,0,7,192,57,255,220   :rem 104
50776 DATA 65,255,194,129,255,193,71 :rem 13
50783 DATA 24,226,56,231,28,0,0 :rem 244
50790 DATA 0,184,85,105,105,85,85   :rem 103
50797 DATA 105,105,85,5,22,22,7 :rem 251
50804 DATA 15,63,62,59,80,148,148   :rem 106
50811 DATA 208,240,252,188,236,15,15 :rem 242
50818 DATA 15,63,63,63,255,255,240  :rem 151
50825 DATA 240,240,252,252,252,255,255 :rem 85
50832 DATA 255,255,255,255,255,255,255 :rem 104
50839 DATA 255,144,83,67,79,82,69   :rem 129
50846 DATA 58,0,144,77,65,71,73 :rem 10
50853 DATA 67,58,0,158,89,79,85 :rem 29
50860 DATA 32,83,65,86,69,68,32 :rem 16
50867 DATA 84,72,69,32,80,82,73 :rem 17
50874 DATA 78,67,69,83,83,32,33 :rem 22
50881 DATA 0,28,66,79,78,85,83      :rem 229
50888 DATA 17,17,157,157,157,157,157 :rem 16
50895 DATA 83,67,79,82,69,17,17 :rem 30
50902 DATA 157,157,157,157,72,73,84 :rem 213
50909 DATA 32,65,78,89,32,75,69 :rem 25
50916 DATA 89,0,28,78,85,77,66      :rem 232
50923 DATA 69,82,32,79,70,32,66 :rem 11
50930 DATA 73,82,68,83,32,40,49 :rem 6
50937 DATA 45,51,41,32,79,82,32 :rem 2
50944 DATA 81,32,84,79,32,81,85 :rem 13
50951 DATA 73,84,0,158,80,79,79 :rem 18
50958 DATA 82,32,80,82,73,78,67 :rem 20
50965 DATA 69,83,83,32,66,76,79 :rem 30
50972 DATA 78,68,69,76,76,0,7       :rem 182
50979 DATA 12,120,7,233,120,8,97:rem 48
50986 DATA 240,7,233,240,7,12,240   :rem 89
50993 DATA 7,12,120,7,233,120,8 :rem 243
51000 DATA 97,120,10,143,120,7,233  :rem 118
51007 DATA 120,8,97,120,7,12,240:rem 29
51014 DATA 28,66,79,78,85,83,58 :rem 23
51021 DATA 0,162,3,160,255,136,208  :rem 126
51028 DATA 253,202,208,248,165,108,208 :rem 87
```

LOAD

Rescue of Blondell

```
51035 DATA 30,198,109,208,26,169,20 :rem 195
51042 DATA 133,109,198,106,165,106,201 :rem 78
51049 DATA 255,208,14,198,107,16,10 :rem 194
51056 DATA 169,1,133,108,169,0,133  :rem 139
51063 DATA 106,133,107,96,169,1,141 :rem 189
51070 DATA 25,208,173,18,208,201,204 :rem 231
51077 DATA 144,36,173,22,208,41,248 :rem 196
51084 DATA 9,0,141,22,208,173,24:rem 34
51091 DATA 208,41,240,9,5,141,24:rem 33
51098 DATA 208,173,22,208,41,239,141 :rem 245
51105 DATA 22,208,169,0,141,18,208  :rem 133
51112 DATA 76,205,199,173,22,208,41 :rem 193
51119 DATA 240,13,62,3,141,22,208   :rem 76
51126 DATA 169,205,141,18,208,173,24 :rem 242
51133 DATA 208,41,240,9,14,141,24   :rem 78
51140 DATA 208,173,22,208,9,16,141  :rem 134
51147 DATA 22,208,173,13,220,41,1   :rem 75
51154 DATA 240,76,206,133,3,208,68  :rem 143
51161 DATA 169,5,141,133,3,173,132  :rem 134
51168 DATA 3,208,30,169,1,141,132   :rem 84
51175 DATA 3,173,52,3,141,249,7 :rem 249
51182 DATA 173,53,3,141,250,7,173   :rem 88
51189 DATA 54,3,141,251,7,169,129   :rem 103
51196 DATA 141,11,212,208,28,173,55 :rem 190
51203 DATA 3,141,249,7,173,56,3 :rem 245
51210 DATA 141,250,7,173,57,3,141   :rem 79
51217 DATA 251,7,169,128,141,11,212 :rem 185
51224 DATA 169,0,141,132,3,76,49:rem 41
51231 DATA 234,76,188,254,162,0,138 :rem 197
51238 DATA 74,168,185,249,7,201,250 :rem 205
51245 DATA 240,117,173,27,212,201,75 :rem 234
51252 DATA 144,43,201,105,176,6,222 :rem 180
51259 DATA 2,208,76,152,200,201,150 :rem 180
```

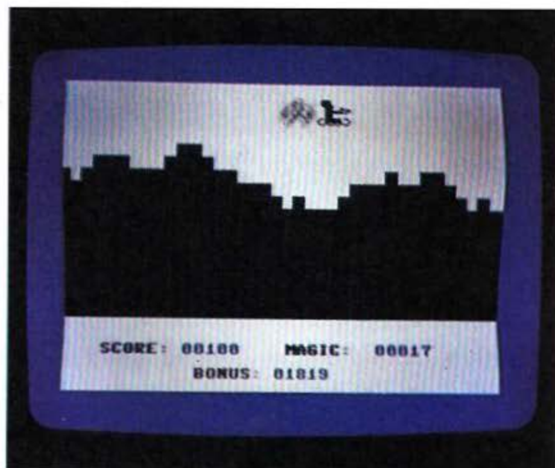
```
51266 DATA 176,13,189,2,208,201,200
                               :rem  187
51273 DATA 240,77,254,2,208,76,152
                               :rem  148
51280 DATA 200,201,190,176,6,222,3
                               :rem  126
51287 DATA 208,76,152,200,254,3,208
                               :rem  194
51294 DATA 76,152,200,189,2,208,205
                               :rem  195
51301 DATA 0,208,240,28,144,11,222
                               :rem  119
51308 DATA 2,208,169,1,157,176,2:rem  45
51315 DATA 76,133,200,189,2,208,201
                               :rem  184
51322 DATA 255,240,27,254,2,208,169
                               :rem  193
51329 DATA 0,157,176,2,189,3,208:rem  50
51336 DATA 205,1,208,240,18,144,6
                               :rem  84
51343 DATA 222,3,208,76,152,200,254
                               :rem  183
51350 DATA 3,208,173,27,212,201,2
                               :rem  75
51357 DATA 176,5,169,1,157,80,3 :rem  3
51364 DATA 232,232,228,21,240,3,76
                               :rem  136
51371 DATA 37,200,96,162,0,189,192
                               :rem  149
51378 DATA 2,208,68,189,80,3,208:rem  59
51385 DATA 3,76,38,201,189,176,2:rem  56
51392 DATA 157,224,2,240,12,189,2
                               :rem  90
51399 DATA 208,56,233,20,157,8,208
                               :rem  155
51406 DATA 76,218,200,189,2,208,24
                               :rem  144
51413 DATA 105,20,157,8,208,32,45
                               :rem  85
51420 DATA 201,141,255,2,173,21,208
                               :rem  173
51427 DATA 13,255,2,141,21,208,189
                               :rem  140
51434 DATA 3,208,157,9,208,169,1:rem  49
51441 DATA 157,192,2,169,56,157,240
                               :rem  202
51448 DATA 2,189,224,2,240,6,222:rem  40
51455 DATA 8,208,76,7,201,254,8 :rem  2
51462 DATA 208,222,240,2,208,26,169
                               :rem  190
51469 DATA 0,157,192,2,157,80,3 :rem  254
51476 DATA 32,45,201,24,105,255,141
                               :rem  184
51483 DATA 255,2,173,21,208,45,255
                               :rem  146
51490 DATA 2,141,21,208,232,232,228
                               :rem  180
51497 DATA 21,208,132,96,138,74,168
                               :rem  212
51504 DATA 169,16,192,0,208,1,96:rem  46
51511 DATA 10,136,76,50,201,173,129
                               :rem  182
51518 DATA 3,208,77,173,128,3,208
                               :rem  100
51525 DATA 1,96,169,1,141,129,3 :rem  250
51532 DATA 173,21,208,9,128,141,21
                               :rem  135
51539 DATA 208,169,129,141,4,212,169
                               :rem  254
51546 DATA 0,141,16,208,169,85,141
                               :rem  145
51553 DATA 131,3,173,1,208,141,15
                               :rem  79
```

LOAD

Rescue of Blondell

```
51560 DATA 208,173,248,7,201,255,208
                               :rem  246
51567 DATA 15,169,1,141,130,3,173
                               :rem  89
51574 DATA 0,208,56,233,19,141,14
                               :rem  90
51581 DATA 208,96,169,0,141,130,3
                               :rem  92
51588 DATA 173,0,208,24,105,19,141
                               :rem  142
51595 DATA 14,208,96,173,130,3,208
                               :rem  149
51602 DATA 26,238,14,208,173,14,208
                               :rem  190
51609 DATA 201,255,208,19,173,16,208
                               :rem  246
51616 DATA 9,128,141,16,208,169,0
                               :rem  96
51623 DATA 141,14,208,76,176,201,206
                               :rem  238
51630 DATA 14,208,206,131,3,173,131
                               :rem  176
51637 DATA 3,141,1,212,208,21,173
                               :rem  79
51644 DATA 21,208,41,127,141,21,208
                               :rem  181
51651 DATA 169,0,141,129,3,141,128
                               :rem  138
51658 DATA 3,169,128,141,4,212,96
                               :rem  103
51665 DATA 169,0,141,13,212,141,12
                               :rem  129
51672 DATA 212,141,8,212,160,0,185
                               :rem  132
51679 DATA 34,199,141,1,212,185,35
                               :rem  154
51686 DATA 199,141,0,212,190,36,199
                               :rem  208
51693 DATA 169,255,133,41,198,41,208
                               :rem  3
51700 DATA 252,202,208,245,200,200,200
                               :rem  55
51707 DATA 192,36,144,223,169,32,141
                               :rem  246
51714 DATA 4,212,169,114,141,13,212
                               :rem  178
51721 DATA 169,17,141,12,212,141,8
                               :rem  134
51728 DATA 212,96,120,169,49,141,20
                               :rem  197
51735 DATA 3,169,234,141,21,3,169
                               :rem  95
51742 DATA 0,141,26,208,169,255,141
                               :rem  191
51749 DATA 13,220,169,0,141,21,208
                               :rem  135
51756 DATA 88,96,32,91,255,162,10
                               :rem  110
51763 DATA 160,3,24,32,240,255,169
                               :rem  143
```

```
51770 DATA 230,160,198,32,30,171,169 :rem 245
51777 DATA 1,141,33,208,165,203,201 :rem 184
51784 DATA 62,208,1,0,201,56,208:rem 38
51791 DATA 4,162,1,208,14,201,59:rem 39
51798 DATA 208,4,162,2,208,6,201:rem 44
51805 DATA 8,208,229,162,3,134,20 :rem 90
51812 DATA 138,10,133,21,76,3,192 :rem 86
51819 DATA 169,0,141,24,212,32,18 :rem 89
51826 DATA 202,173,24,208,41,240,9 :rem 139
51833 DATA 5,141,24,208,169,147,32 :rem 146
51840 DATA 210,255,162,10,160,8,24 :rem 130
51847 DATA 32,240,255,165,113,164,114 :rem 36
51854 DATA 32,30,171,165,106,133,253 :rem 238
51861 DATA 165,107,133,254,169,19,141 :rem 45
51868 DATA 249,207,32,120,193,162,18 :rem 253
51875 DATA 160,12,24,32,240,255,169 :rem 195
51882 DATA 194,160,198,32,30,171,169 :rem 2
51889 DATA 0,141,33,208,165,106,24 :rem 144
51896 DATA 109,168,2,133,253,165,107 :rem 255
51903 DATA 109,169,2,133,254,169,99 :rem 212
51910 DATA 141,249,207,32,120,193,36 :rem 238
51917 DATA 203,112,252,76,46,202,169 :rem 246
51924 DATA 8,141,3,212,169,20,141 :rem 85
51931 DATA 5,212,169,240,141,6,212 :rem 134
51938 DATA 169,9,141,15,212,169,19 :rem 160
51945 DATA 141,4,212,160,255,140,1 :rem 130
51952 DATA 212,152,160,3,202,208,253 :rem 231
51959 DATA 136,208,250,168,140,39,208 :rem 52
51966 DATA 136,208,237,140,4,212,96 :rem 202
```





Accordatore per chitarra

di **C. Visco**
trad. ed adatt.
di **F. Sarcina**

Con questo breve programma, veloce da caricare e subito pronto all'uso, ti offriamo la possibilità di accordare perfettamente la tua chitarra a 6 o a 12 corde, con l'ausilio del tuo calcolatore. Il programma, che viene presentato in due distinte versioni, per C64 e per C16, ti permette di ascoltare la giusta intonazione di una corda semplicemente premendo uno di questi tasti, in ordine dal suono più grave a quello più acuto:

"E" per la sesta corda (MI basso)
"A" per la quinta corda (LA)
"D" per la quarta corda (RE)
"G" per la terza corda (SOL)
"B" per la seconda corda (SI)
CTRL + "E" per la prima corda (MI cantino).

Se possiedi una chitarra a 12 corde, puoi ottenere l'altezza di una corda della serie più acuta semplicemente premendo il tasto SHIFT (o, naturalmente, SHIFT LOCK) insieme a quelli già elencati.

I tasti da premere per produrre i vari suoni non sono presi a caso: i più esperti si saranno infatti già accorti che questi corrispondono alla notazione in uso nei paesi anglosassoni. Per chi non la conoscesse ecco la sequenza completa delle 7 note:

C in notazione inglese= DO in notazione italiana
D = RE
E = MI
F = FA
G = SOL
A = LA
B = SI

Non ci resta che augurarti un buon concerto, sicuri di avere reso un po' più facile e rapida la noiosa operazione dell'accordatura.

Accordatore per chitarra versione per C64

```
8 DIMHI(12),LO(12),NO$(12)            :rem  115
10 PRINTCHR$(14)"{CLR}{ 8 DES}ACCORDATORE
    PER CHITARRA":FORT=0TO2000:NEXT
                                      :rem    0
15 PRINT"{CLR}{ 4 GIU'}{ 6 DES}USA I TAST
   I E/A/D/G/B/CTRL-E"                :rem  216
16 PRINT"{ 3 GIU'}{ 7 DES}PER UNA CHITARR
   A 6 CORDE."                        :rem   32
17 PRINT"{ 3 GIU'}{ 7 DES}PREMI IL TASTO
   SHIFT-LOCK"                        :rem  168
18 PRINT"{ 3 GIU'}{ 7 DES}PER LA SERIE DI
    CORDE PIU'"                       :rem   51
```

```
19 PRINT"{ 3 GIU'}{ 5 DES}ACUTA DELLA CHI
   TARRA 12 CORDE."                 :rem  27
20 S=54272:FORT=0TO23:POKES+T,0:NEXT:POKE
   S+24,12:POKES+5,17:POKES+6,243
                                    :rem  94
70 FORT=1TO11:READHI,LO:HI(T)=HI:LO(T)=LO
   :NEXTT                           :rem  32
80 GETA$:IFA$=""THEN80              :rem 243
90 A=-(A$="E")-2*(A$="A")-3*(A$="D")-4*(A
   $="G")-5*(A$="B")-6*(A$=CHR$(5))
                                    :rem 171
95 A=A-7*(A$="E")-8*(A$="A")        :rem  62
97 A=A-9*(A$="D")-10*(A$="G")-11*(A$="B")
                                    :rem  34
100 POKES,LO(A):POKES+1,HI(A)       :rem 211
150 POKES+4,17:FORI=0TO2000:NEXTI:POKES+4
    ,16                             :rem 183
175 POKE198,0:GOTO80                :rem 165
200 DATA10,143,14,24,18,209,25,30,31,165,
    42,62,21,31,28,49,37,162,50,60,31,165
                                    :rem  20
```

Accordatore per chitarra versione per C16

```
10 DIMNO(12)                        :rem 140
20 PRINTCHR$(14)"{CLR}{ 8 DES}ACCORDATORE
   PER CHITARRA":FORT=0TO2000:NEXT
                                    :rem   1
```




```
30 PRINT"{CLR}{ 4 GIU'}{ 6 DES}USA I TAST
   I E/A/D/G/B/CTRL-E"              :rem 213
40 PRINT"{ 3 GIU'}{ 7 DES}PER UNA CHITARR
   A 6 CORDE."                      :rem  29
50 PRINT"{ 3 GIU'}{ 7 DES}PREMI IL TASTO
   SHIFT LOCK"                      :rem 120
60 PRINT"{ 3 GIU'}{ 7 DES}PER LA SERIE DI
   CORDE PIU'"                      :rem  48
70 PRINT"{ 3 GIU'}{ 5 DES}ACUTA DELLA CHI
   TARRA 12 CORDE."                 :rem  24
80 FORT=0TO12:READNO:NO(T)=NO:NEXTT
                                    :rem  44
90 GETA$:IFA$=""THEN90              :rem 245
100 A=-(A$="E")+2*-(A$="A")+3*-(A$="D")+4
    *-(A$="G")+5*-(A$="B")+6*-(A$=CHR$(5)
    )                               :rem 170
110 A=A+7*-(A$="E")+8*-(A$="A")     :rem 184
120 A=A+9*-(A$="D")+10*-(A$="G")+11*-(A$=
    "B")                            :rem 198
130 IFA=0THEN90                     :rem 105
140 VOL7:SOUND1,NO(A),180           :rem 209
150 GOTO90                          :rem  56
160 DATA0,345,516,643,739,798,854,685,770
    ,834,881,798,854                :rem 124
```

DISTRIBUZIONE: un anello strategico tra fornitori e utenti finali.



SoftDealer: l'unica rivista dedicata al mondo della distribuzione di home e personal computer, software e libri.

SoftDealer: è il primo mensile italiano per chi vende home e personal computer, software, libri e accessori di informatica. Uno strumento indispensabile per computershop, librerie e cartolibrerie. Su SoftDealer, inviato solo in abbonamento, troverete tutte le notizie più utili, recensioni, test, prove di software, articoli di marketing, classifiche dei programmi e dei libri più venduti in Italia.

SOFT DEALER è un'idea

PROGRAM editrice

Uffici Tecnici e Commerciali:
20125 Milano - Via M. Gioia, 114
Tel. (02) 603415-680851

ABBONATI A SoftDealer: Un modesto investimento per essere informati sul futuro

Ritagliare e spedire in busta chiusa a:

PROGRAM EDITRICE, Via M. Gioia 114, 20125 MILANO

☐ Desidero sottoscrivere un abbonamento annuo (10 numeri L. 25.000) alla rivista SOFT DEALER

☐ Allego assegno di L. 25.000 intestao a PROGRAM EDITRICE

n° .... Banca ....

☐ Ho effettuato pagamento tramite vaglia postale.

☐ Ho effettuato versamento su ccp n. 34510206 intestato a Program-Milano e allego fotocopia ricevuta.

NOME ....

COGNOME ....

AZIENDA ....

VIA .... N° ....

C.A.P. .... CITTÀ ....

PRINT

Recensioni Libri

I libri possono essere ordinati utilizzando il coupon pubblicato a pag. 82

DATA BASE: L'UTILE E IL DILETTEVOLE
di **N. Freestone**
Gruppo Editoriale Jackson

Sono sempre in maggior numero le persone che, dopo aver acquistato un computer più che altro per curiosità o per disporre dei videogame più all'avanguardia, si accorgono dopo un certo tempo che questa prodigiosa macchinetta può essere messa in grado di fornire un valido aiuto per la gestione dei propri piccoli affari.
L'applicazione di interesse più generale per chi decidesse di utilizzare proficuamente il proprio computer è senz'altro quella della memorizzazione di informazioni. Il Database, o archivio elettronico, è un mezzo per fare in modo che il nostro computer "inghiotta" tutti i dati che gli forniamo in modo disordinato e li memorizzi, ordinandoli secondo uno schema da noi prefissato. Il Database è un programma che deve essere "elastico" per adattarsi a qualunque tipo di dato si desidera memorizzare, dall'archiviazione delle fatture di un'azienda alle ricette di cucina per la massaia. A causa di questo aspetto polimorfo, in generale un buon Database è un programma molto complesso, che richiede almeno un minimo di esperienza per essere sfruttato in tutte le sue potenzialità.
Questo libro è una guida per chi, dotato di una conoscenza sufficiente del linguaggio BASIC, desidera crearsi un programma di gestione dati "su misura": vengono da principio analizzati i tipi di dati che possono essere elaborati dal computer ed il modo migliore di organizzarli, per ottenere allo stesso tempo affidabilità e velocità di esecuzione del programma.
Si passa quindi ad esaminare tutti i diversi tipi di file che è possibile usare all'interno del Database, con i vantaggi e gli svantaggi legati ad ognuno di essi, ed i metodi di ordinamento dei dati. Questi ultimi rappresentano, insieme alle tecniche di ricerca, il cuore di un Database, e sono qui chiaramente spiegati con l'ausilio di esempi pratici, consistenti in routine BASIC che eseguono i principali algoritmi di ordinamento e di ricerca di un record. Si passa quindi alle applicazioni pratiche, consistenti in analisi dei problemi tipici per un sistema di conti bancari, un'agenda, un controllo sulla giacenza, cataloghi ed indirizzi, elenco dei debitori e libro paga. È molto facile, da questi esempi, trarre spunti e consigli, da utilizzare nel proprio specifico campo di applicazione, che permettono di iniziare subito con la stesura del proprio programma.
Il libro fornisce tutte le cognizioni tecniche necessarie per la costruzione di un Database, e rappresenta un prezioso aiuto per chi non intende ricorrere a programmi "preconfezionati", che, appunto per questo essere universali, raramente rispondono appieno alle singole esigenze.

IL LIBRO DEI GIOCHI PER COMMODORE 16 E PLUS/4
di **A. Borra** e **M. Cristuib Grizzi**
ediz. ***J. soft***

Recensione a cura di
Lucio Bragagnolo

La firma di chi scrive può apparire insolita per il lettore di SUPERCOMMODORE: in effetti, non potendo ovviamente l'ottimo Mauro recensire un... suo libro, la redazione di SUPERSINC si è prestata volentieri al gradito compito, anche

per approfittare dell'occasione per fare al nostro collega i migliori auguri affinché il frutto della sua fatica riscuota il (meritato) successo presso i lettori. E ora, con la coscienza a posto, possiamo finalmente chiudere questo preambolo per entrare nel vivo del discorso.

I due nuovi computer della Commodore sono finalmente dotati di un linguaggio BASIC evoluto, tale da permettere all'utente la realizzazione di programmi anche sofisticati, senza costringerlo ad "annegare" in un mare di PEEK e POKE.

Tuttavia, il problema principale che si pone sempre alla comparsa sul mercato di nuovi modelli di computer è la – seppur inizialmente – difficile reperibilità di programmi da far "girare" subito sulla nuova macchina.

La J. soft, editrice della rivista che state leggendo, ha ora pubblicato questo nuovo volume dedicato ai possessori del Commodore 16 e Plus/4: tutti i programmi contenuti nel libro sono infatti perfettamente compatibili con entrambi i computer.

Questa è la prima raccolta attualmente disponibile in Italia di giochi per i nuovi computer Commodore, ed appartiene a quella generazione appena nata di nuovi libri "informatici", sviluppati proprio per l'utilizzo diretto con il computer.

La struttura stessa del libro, infatti, è organizzata in modo tale da poter fornire – tramite la cassetta allegata – i programmi già pronti per l'uso e, al contempo, offrire insieme alla descrizione completa di tutti i programmi anche i loro listati su carta.

Una particolare nota di merito è senz'altro da attribuire a questi ultimi, che sono stati realizzati (ormai i lettori di SUPERCOMMODORE ci saranno abituati!) con grande cura e con l'utilizzo di una particolare codifica la quale – unitamente all'uso di una stampante "letter-quality" – li rende estremamente chiari e leggibili anche per il neofita.

Il libro si propone infatti come mezzo per ottenere un immediato divertimento con il proprio computer, permettendo contemporaneamente l'acquisizione delle tecniche utilizzate per la realizzazione dei giochi stessi.

I venti programmi proposti spaziano dall'"Adventure game" all'"Arcade" ed al simulatore di volo, offrendo una panoramica completa su tutti i tipi di videogioco più interessanti e divertenti; e la veste editoriale è quella ormai "classica" della produzione J. soft, con una comoda rilegatura ad anelli.

7 NOTE BIT, OVVERO LA MUSICA A PORTATA DI MANO

Il Gruppo Editoriale Jackson, in collaborazione con la SIEL Software Division, ha realizzato un'opera destinata a fare storia nell'alfabetizzazione informatica italiana.

La pubblicazione quattordicinale "7 NOTE BIT" da poco avviata, rappresenta infatti quella che si può definire come la più completa, tecnicamente sofisticata, e più interattiva pubblicazione su supporto magnetico/cartaceo fino ad ora realizzata in Italia ed all'estero: "7 NOTE BIT" permette infatti di conoscere la musica, imparare a suonare uno strumento e, contemporaneamente, imparare a programmare in BASIC.



"7 NOTE BIT" consiste in un fascicolo di 32 pagine a colori, ed in una cassetta contenente il software da caricare sul proprio C64. Con il primo numero, viene offerta in omaggio una eccezionale tastiera musicale da applicare sopra il computer, che permette di eseguire comodamente tutti gli esercizi che via via vengono proposti dal corso.

"7 NOTE BIT" è infatti un vero e proprio corso, suddiviso in tre sezioni integrate, e dotato di soluzioni tecniche e grafiche mai viste fino ad ora su prodotti similari (basti, come unico esempio, la presenza di una routine – ovviamente in l/m – che permette al C64 di mantenere attivo il video durante il caricamento "turbo" da nastro, e nel contempo suonare una piacevole musica per intrattenere l'utente durante il caricamento stesso!). Le tre sezioni del corso riguardano musica, tastiera ed informatica musicale: la prima comprende nozioni generali sulla musica, gli stili e gli strumenti, mentre la seconda sezione consiste nella parte pratica che insegna a leggere la musica ed a eseguirla, con un'intensa interazione computer/utente, tale da poter imparare divertendosi. Infine, la terza sezione introduce in modo estremamente chiaro le nozioni di programmazione musicale in BASIC: al termine del corso l'utente sarà perfettamente in grado di programmare da sé il computer, per ottenere gli effetti musicali desiderati.

"7 NOTE BIT", in edicola ogni 14 giorni a L. 10000, è disponibile anche in abbonamento (15 fascicoli L. 145000).

I libri italiani vengono offerti con il 10% di sconto sul prezzo di copertina

input·output

Piccoli annunci

Cambio programmi su cassetta come: Decathlon, Leopard, Giancarlo il giardiniere e molti altri. Programmi di utility: Turbo Tape, che dimezza la durata del caricamento, 80 Colonne che amplia la memoria. Giacomo Marconi - Via 7 Ponti, 58 - 52040 Quadrata (AR) - Tel 364075

Vendo cassetta o disk contenente tre giochi e quattro utility, cioè: Protector 2 - Indian - Fort Apache - Pipes - Sea Sound - Pet Speed - Monitor, tutto a L. 40.000. Scrivere o telefonare. Salvatore Purgato - Corso Europa, 26 - 81030 S. Marcellino (CE) - Tel. 081/8121276

Per Commodore 64 vendo programma Totocalcio che elabora sistemi integrali condizionati al numero di segni 1-X-2. Funziona anche in stampante. Per informazioni scrivere a: Sebastiano Caramagno - Contrada Cipollazzo - 96011 Augusta (SR)

Vendo le seguenti cassette per Atari VCS 2600 a metà prezzo: Dragster, Oink Football, Enduro, Raiders of the Lost Arck, Ghost Manor/Spikès, Donkey Kong, Pitfall, Tennis, Spider Man, Space Shuttle. Cristaino Cravanzola Via Prof. Oliva, 16 - 12011 Borgo S. Dalmazzo (CN) - Tel. 0171/769659

Compro o cambio programmi (in maggioranza giochi) per VIC 20 inespanso o espanso max. 16 Kbyte. Vendo cartridge: Road Race, Poker, The Sky is falling e altre. Compro stampante per VIC 20 se a buon prezzo. Scrivere a: Leonardo Pesco - Via S. Nicastro, 4 - 91026 Mazara del Vallo - 0923/945623

Vendo Intellevision più joystick, 9 cassette (Buzz Bombers, Lock'n'chase, Burger Tima, Pitfall, Shark Shark, Micro Surgeon, Vectron, Space Armada, Poker e Blackjack) a L. 350.000; valore commerciale L. 800.000. Roberto Festini - Via Astinizza, 96 - 14055 Costigliole d'Asti (AT) - Tel. 0141/966181

Vendo e cambio per C64 oltre 1.000 programmi di utility e giochi (praticamente quasi tutto il software attualmente disponibile per C 64 a prezzi ecezionali. Inviare L. 1.000 per lista. Franco Billi - Corso Milano, 36 - 14100 Asti - Tel. 0141/55182

Vendo programmi per CBM 64 a sole L. 7.000 cadauno. Invio liste gratis. Mandatemi le vostre richieste. Spedizione in contrassegno. Sono anche disposto a scambio di software. Carlo Fiorletta - Via S. Silvestro, 2 - 03011 Alatri (FR)

Cambio programmi ed utility per C64 (circa 400). Inoltre cambio telescopio professionale più cavalletto ed accessori con floppy per CBM 64. Luciano Viola - Via Scassa, 27 - 71036 Lucerna (FG) - Tel. 942255

Vendo o cambio software per CBM 64. Dispongo di molti programmi di utilità, gestionali e giochi. Novità, prezzi modici. Dino Marocchi - Via Marconi, 302 65100 Pescara - Tel. 085/68352

Per C64 vendo, cambio nuovissimi programmi ed utility a prezzi irrisori: Soccer, Sam Reciter, Solo Flight, Jet Set Willy, Pole Position, Pipeline, Indiana Jones, Pit Stop, One On One, Baseball. Tutto a L. 35.000. Rispondo a tutti. Francesco Donia - Via Ugo Soli, 9 - 90128 Palermo - Tel. 091/426143

Per C64 cambio e vendo giochi come Zaxxon, Baseball; utility come Sam Reciter... Cerco Ultra Copy o similari. Effettuo cambi e vendite anche per VIC 20 espanso e non. Pierangelo Galizia - Via Appia SS 7, 3 - 85050 Baragiano (PT) - Tel. 0971/993331

Vendo CBM 64 piu registratore 1530, vari giochi ed utility. Usato pochissimo L. 600.000. Rolando Carlini - Via Stoccolma, 51 - 47037 Rimini - Tel. 0541/33868

Per C64 vendo, cambio molti giochi fra cui: Pitfall II, Falcon Patrol I e 2, Popeye, Dr. J. and L. Bird, Magic Desk, Decathlon, Forbidden Forest, Widow's Revenge, Mr. Robot; tutti in Turbotape a
L. 3.000. Telefonare e inviare liste. Annuncio sempre valido. Fabio Massabò - Via Roma, 20 - 18039 Ventimiglia - Tel. 0184/351829

Vendo per C64 cassetta contenente i seguenti programmi: Tool 64, Simons' BASIC, FORTH 64, Pascal, The Bells, Skislalom, Pool 64 a sole L. 15.000. Telefonare o scrivere a: Paolo Schirra - Via Taranto, 4 - 09170 Oristano - Tel. 0783/259666

VIC 20 Club sta realizzando per gli aderenti espansione 16/24 Kbyte. Si accettano richieste esterne. Flavio Stanchi - Via Sparta, 26/22 Sestri P. - 16153 Genova - Tel. 010/602435

Vendo, cambio moltissimo software per C64, a scelta su disco o nastro. Inviare lista o richieste. Annuncio sempre valido. Paolo Lalli - Via Castellano - 64030 Scorrano (TE) - Tel. 0861/66437

Eccezionale! Per CBM 64 vendo cassetta o disco (più L. 8.000) con i seguenti programmi a sole L. 40.000: Decathlon, International Soccer, Jungle Hunt, Biliardo, Blue Moon, King Kong, Jambreaker e Pac Man. Inoltre, per tutti gli acquirenti, la versione originale di Turbo Tape in omaggio. Sono anche disposto a scambiarli con Sam più Simons' BASIC e Magic Desk su nastro. Annuncio sempre valido. Scrivere a: Gaetano Sandovalli - Via C.le Massaia, 53/a - 90142 Palermo

Vendo più di 30 programmi (giochi ed utility) per VIC 20 a L. 5.000 ciascuno più uno in regalo. Telefonare solo ore pranzo. Fabio Benedetti - Piazza Istria, 28/6 - 54100 Massa - Tel. 330437

Vendo VIC 20 più registratore, joystick, corso BASIC, cassette videogiochi, riviste, manuale, tutto a L. 300.000 non trattabili. Telefonare dopo le ore 20.00 a: Claudio Pucci - Via F. Menzocchi, 1 - 47100 Forlì - Tel. 0543/24027

Vendo oltre 500 programmi per C64, giochi ed utility. Il prezzo è di L. 5.000 per ogni programma; diminuisce per l'acquisto di una certa entità. Antonio Licata - Via A. Zacco, 38 - 95125 Catania - Tel. 095 330531

Compro programmi per VIC 20 inespanso o con 16 Kbyte. Inviare liste programmi. Giampaolo Gori - Via G. Puccini, 33 - 57100 Livorno - Tel. 0586/401698

Compro utility e giochi per C64. Inviare liste e prezzi per visione. Sono interessato poi a Turbo Tape e Simons' BASIC con istruzioni scritte in italiano. Antonio Noto c/o Derito - Viale XX Settembre, 21 - 54033 Carrara (MS)

Cambio programmi per C64. Ho più di 100 programmi sia su disco che cassetta. Inviate la lista dei vostri programmi e vi invierò la mia. Graziano Camia - Via Augusta, 100 - 39100 Bolzano - Tel. 0471 44874

Vendo VIC 20 più registratore C2N, memorie 16 e 8 Kbyte, 3 cartridge e parecchi giochi, volumi e cassette BASIC 1 2, libro "Il Commodore VIC 20" a L. 400.000. Renzo Monteccai - Via Mazzini, 11 - 20090 Vimodrone (MI) - Tel. 02 2504966

Cambio, vendo molto software espanso e inespanso per VIC 20, su nastro, dischetti, listati. Offerte vantaggiosissime. Inviate la vostra lista a: Gregorio Lena - Viale Silvani, 3/2 - 40122 Bologna - Tel. 051/551178

Cerco per Commodore 64 giochi su cassetta a prezzo non superiore a L. 4.000 ognuno. Ecco i nomi: Decathlon, Zaxxon, Pole Position, Calcio. Marcello Grandi - Via Dante Alighieri, 2 - 50064 Incisa (FI) - Tel. 055/8335003

Scambio e vendo programmi, giochi ed utility per C64. Chi vuole contattarmi può telefonare o scrivere inviando la lista dei programmi. Eros Cattaneo - Viale Montenero, 78 - 20135 Milano - Tel. 02/599280

Compro espansioni 3 Kbyte e 8 Kbyte per VIC 20, sólo se buon prezzo. Alberto Cacciari - Via S. Allende, 15 - 40139 Bologna

Cerco il libro "Alla scoperta del VIC 20" edito da Editoriale Jackson. Chi ne è in possesso e lo vuole vendere, mi contatti: offro L. 10.000. Vincenzo Di quattro - Via La Masa, 15 - 97100 Ragusa

VIC 20 Software Club - Se avete problemi per avere programmi, scambiare esperienze, ecc., associatevi a noi. L'iscrizione è gratuita e garantiamo risposta a tutti. Massima serietà. VIC Soft Club c/o Cinzia Ceccarini - Via Di Vittorio, 10 - 58022 Follonica (GR)

Vendo e cambio giochi ed utilità per CBM 64 a prezzi al di sotto dell'economicità. Massima serietà. Invio lista, attendo la vostra. Scrivo a tutti quanti. Felice Vacchiano - Via Corpo di Cristo - 80033 Cicciano (NA) - Tel. 081/8268772

Vendo VIC 20 più 4 cassette giochi a L. 230.000. Valerio Battaglini - Via Argentieri Vico 1, 3 - 86036 Montenero di Bisaccia (CB) - Tel. 966231

Vendo listati per: Apple, TI 99/4A, Sinclair Spectrum 16/48 Kbyte, C64 e VIC 20. Oppure scambio con il VIC 20. Scrivere a: Armando Mendolicchio - Via Gino Acquaviva, 27 - 71100 Foggia

Compro per VIC 20 espansioni di memoria da 3-8-16-24 Kbyte e programmi di geometria e trigonometria a prezzi vantaggiosi. Compro anche programmi di fisica. Flavio Franco - Via Monte delle Gioie, 34 - 00199 Roma - Tel. 06/8391353

Vendo giochi per CBM 64 riportati, in parte, da cartuccia a cassetta. Tra essi dispongo di: Pole Position, Forbidden Forest, Le Mans, Fort Apocalypse, Popeye, Slalom, Froggee e tanti altri, tutti eccezionali. Con essi è fornito il Turbo Tape che li aziona. I giochi sono forniti a L. 5.000 cadauno - Raffaele Vitale - Via Vesuvio, 14 - 80058 Torre Annunziata (NA) - Tel. 081/2618602

Cambio circa 1.500 programmi per C64 con materiale radioamatori - CB - ricevitori e TV usato. Telefonare ore pasti. Giuseppe Borracci - Via Mameli, 15 - 33100 Udine - Tel. 0432/291665

Vendo giochi su cassetta per C64 e VIC 20. Offro a L. 20.000 cassetta con i seguenti giochi per CBM 64: Kongo Bongo, Aztec, Decathlon, Soccer, Popeye, Crazy Kong. Scrivete a: Luca Nardini - Via Colle Perino - 00049 Velletri (RM)

Cambio gioco elettronico "Dribblingtronic" con espansione 16 Kbyte per VIC 20. Marzio Colleoni - Via Stazione, 22 - 11020 Hone (AO)

Compro bobine metalliche vuote per registratore professionale (diametro cm 24). Solo Milano e provincia. Maurizio Violi - Via Molinetto di Lor. - 20094 Corsico (MI) - Tel. 02/4407292

Vendo per Commodore 64 vari giochi da L. 1.500 a L. 5.000 (Zaxxon, Donkey Kong...) disponibili su cassetta o disco. Vendo per C64 Turbo Tape a L. 4.000. Allego ad ognuno un elenco (gratuito) dei giochi in mio possesso. Per informazioni - Marco Stefanono - Via Marconi, 17 - 21052 Busto Arsizio (VA) - Tel. 0331/639943

input·output

Piccoli annunci

Compro, vendo, cambio programmi per C64. Cerco sopratutto programmi di studio, gestionali, utilities a poco prezzo. Vendo cassette con programmi registrati (solo giochi) a L. 10.000-15.000 cadauno. Giorgio Ciavaglia - Via Vetulonia, 33 - 00183 Roma - Tel. 06/7587969

Per C64 vendo, cambio, programmi di giochi, utility, musicali, grafica in alta risoluzione, matematica, didattici. Richiedere lista o informazioni a: Mario Massimo Caputo - Via Roma, 19 - 81030 Teverola (CE) - Tel. 081/8118104

Vendo, per VIC 20 inespanso, 4 splendidi giochi su cassetta all'eccezionale prezzo di L. 60.000. Un'occasione da non perdere! Pagamento tramite vaglia postale. Mariano Campanelli - Via Dott. L. Paolucci, 8 - 63033 Cento Buchi (AP)

Vendo VIC 20 più registratore, joystick, 60 programmi (di cui 33 in LM), alimentatore e cavi, molte riviste di computer, manuali e libro in italiano, ancora in garanzia, imballaggi originali, il tutto a sole L. 350.000. Marcello Galanti - Via Torrenova, 134 - 00133 Roma

Vendo (causa passaggio a sistema superiore) VIC 20 più registratore dedicato, joystick, 3 manuali, 1 cassetta con 40 stupendi giochi, molte riviste e più di 300 programmi. Il tutto in ottime condizioni a L. 380.000. Giovanni Azara - Via Peone, 3 - 20089 Rozzano (MI) - Tel. 02/8240807

Vendo, cambio listati di programmi per VIC 20. I prezzi vanno da L. 500 a L. 2.500. Vendo programmi (o cambio con altri per VIC 20) per C64, ZX81 16/48 Kbyte, Apple, TI 99/4A; MPF II, Oric 1. Nonostante tutto mi occupo solo di listati per VIC 20. Eraldo Caprara - Via Ortelle, 23 - 02019 Supino (FR)

Vendo, cambio programmi per C64. Invio lista gratuita. Inviare lista per cambi. Carlo Sciascia - Via E. Raggio, 32/2 - 15067 Novi Ligure - Tel. 0143/70089

Per C64 vorrei acquistare software giochi e utility, qualsiasi tipo. Per vendite inviare lista, prezzi, caratteristiche, ecc. Senza impegno di acquisto. Programmi su nastro o listati. Marco Lucchini - Via Marengo - 46045 Marmirolo (MN)

Vendo base Intellevision con 13 ottime cassette (3 Imagic) a sole L. 250.000 causa acquisto computer. Il tutto in perfette condizioni. Paolo Mugliari - Via T. Gulli, 19 - 20147 Milano - Tel. 02/4046559

Per C64 vendo o cambio tantissimi giochi a prezzi eccezionali, ad esempio: Pole Position, Simons' BASIC L. 10.000, Defender, Dig Dug L. 8.000. Tutti registrati con Turbo Tape. Scrivete per invio listino. Telefonare per scambi. Luca Castiglioni - Via Milano, 69 - 22070 Bregnano (CO) - Tel. 031/772736

Compro, vendo, cambio programmi in linguaggio macchina per C64 (Manic Miner, Donkey Kong, Pole Position, Pit Stop, ecc.). Scrivere o telefonare ore pasti. Emanuele Rienzi - Via Capo Poro, 45 - 00122 Roma Lido - Tel. 06/5602133

Vendo 6 programmi su cassetta con Turbo Tape a L. 35.000: Basket, H.E.R.O., Fort Apocalypse, Baseball, Star Trek originali. Invio liste. Roberto Costagliolo - Via G. Ricci Curbastro, 34 - 00149 Roma - Tel. 06/5589365

Ho fondato un club privato per VIC 20, e se qualcuno vuole mandarmi i propri programmi, io ne invierò degli altri. Cerco anche espansione 16 Kbyte a prezzi intorno alle L. 50.000. Assicuro massima serietà. Valentino De Diana - Via Brodevin - 32040 Lozzo di Cadore (BL)

Vendo VIC 20 più registratore C2N, joystick, BASIC parte 1, 80 giochi in cassetta, 1 cartridge, libro "Alla scoperta del VIC 20" a L. 300.000 non trattabili. Claudio Spadazzi - Via Tito Papirio, 14 - 47100 Forlì - Tel. 0543/35602

Vendo cassetta contenente 11 giochi originali Commodore a L. 50.000 (Mundial Soccer, Pole Position, Pac Man, Sky Race ...). Inoltre vendo cartucce: Jupiter Lander a L. 20.000 e Radar Rat Race a L. 25.000. Avviso sempre valido! Marcello Caputo - Via Filangeri, 1 - 71100 Foggia - Tel. 0681/25072

Vendo consolle Intellevision più 14 cassette giochi a L. 500.000. Vendo color computer Laser 200 9 colori, 16 Kbyte, espandibile fino a 64 Kbyte, sonoro più una cassetta dimostrazioni, e manuale d'uso a L. 100.000. Vincenzo Roscigno - Via Roma, 52 - 84091 Battipaglia (SA) - Tel. 0828/21567

Vendo programmi su listato per VIC 20 a L. 500 cadauno più L. 1.000 per spese postali. Chi è interessato scriva su lettera nome, cognome, indirizzo, città, CAP, numero telefonico e i listati richiesti. Inserire anche prezzo e numero dei listati. Cera Antonio - Via Volta - 13044 Crescentino (VC) - Tel. 0161/843055

Vendo VIC 20 come nuovo più registratore, cartuccia 16 Kbyte, joystick, introduzione al BASIC 1 e numerosi listati a L. 400.000. Enzo Cati - Viale Stazione, 25 - 12032 Barge (CN) - Tel. 0175/36762

Per Commodore 64 vendo in blocco: 2 cartucce (Radar Rat Race e Visible Solar System), cassetta Scacchi Grand Master, introduzione al BASIC parte 1 con manuale e 2 cassette a L. 120.000 non trattabili. Scrivere o telefonare a: Gino Gelsomino - Via Belvedere, 10 - 10011 Agliè (TO) - Tel. 0124/33213

Vendo per VIC 20: cabinet a 6 slot model VIC 20 a L. 200.000, S.E. a L. 35.000, Sintetizzatore vocale à L. 140.000, 16 Kbyte a L. 80.000 e VIC 20 a L. 110.000. Telefonare dopo le 14. Beppe De Vanna - Via Fontanelle, 8 - 70057 Palese (BA)

Vendo VIC 20 più 16 Kbyte RAM, alta risoluzione, 2 cartridge (Poker, Sistema Solare Visibile), 100 programmi su cassette. Il tutto a L. 290.000 e in ottimo stato. Roberto Trezzi - Via Don E. Cattaneo, 5 - 20025 Legnano (MI) - Tel. 0331/544017

Eccezionale! Vendo software per VIC 20 inespanso a prezzi fantastici. Giochi come "Trout", "Konghy", "Defender" praticamente gratis. Per la lista e informazioni scrivere allegando L. 500 a: Bruno Sereni - Via Dosso, 14 - 26034 Piadena (CR)

Cambio o vendo 300 programmi per VIC 20 in LM e RLM. Scrivetemi, vi invierò il mio catalogo con relative istruzioni. Carlo Vincenzi - Via Resistenza, 26 - 41033 Concordia (MO) - Tel. 54325

Cambio software per C64. Annuncio sempre valido. Domenico Rizio - Via Capestrano, 20 - 66013 Chieti Scalo - Tel. 0871/584239

Per C64 vendo, cambio programmi su disco. Possiedo utility, gestionali, molti videogames da bar. Chiedere elenco allegando busta affrancata per la risposta. Anna Di Ruscio - Via Marconi, 27 - 63010 Montefiore dell'Aso (AP)

Vendo Commodore VIC 20 più registratore Maxtron, espansione 3/8/16 Kbyte, 40 giochi di ogni tipo a L. 400.000. Fabio Alpiovezza - Via Ponale, 6 - 20162 Milano - Tel. 02/6433182

Per CBM 64 vendo e cambio programmi su cassetta e floppy: Turbo Tape, Simons' BASIC, Frogger, Basket, Calcio, ecc. Scrivere o telefonare per invio lista di circa 1.000 tra giochi ed utility. Molti in omaggio. Francesco Petriccione - Via Venezia, 43 - 81022 Casagiove (CE) - Tel. 0823/466418

Per Commdore 64 e Sinclair Spectrum disponendo di un notevole archivio software vendo a L. 10.000 cassetta con 5 giochi a scelta. Anche su disco. Annuncio sempre valido. Scrivere a: Ivano Parbuono - Via A. di Cambio, 4 - 37138 Verona - Tel. 045/568649

Compro, cambio, vendo software per Commodore 64. Molti giochi ed utility, richiedere la lista allegando L. 1.000 per spese postali. Cesare Giarda - Via Viscerei, 33 - 28065 Cerano (NO)

Vendo, cambio software per VIC 20 su cassetta. Invio la mia lista di programmi a chi mi invia la propria. Alcuni dei programmi sono disponibili anche su listato. Ivano Modenesi - Viale Sicilia, 55 - 27100 Pavia - Tel. 0382/472142

Vendo VIC 20 in perfette condizioni con imballaggio a L. 150.000 più spese postali, compresa cartuccia Sargon Chess II. Telefonare a: Lauro Romanzin - Via Candotti, 33/1 - 33033 Codroipo (UD) - Tel. 0432/906352

Vendo CBM 64, datassette, 2 joystick Spectravideo II, cassetta Drig Daclaton, garanzia italiana GBC, istruzioni italiano, ancora in scatola originale. Acquistato Ottobre 1984 vendo, per passaggio a videogioco Intellevision, a L. 600.000. Paolo Mazzei - Via Tommaso Pini, 45 - 50047 Prato - Tel. 0574/596018

Vendo computer Luky, da inserire nel videogames Intellevision. A partire da L. 200.000, compresa cassetta Scooby-Doo Maze Chase. Telefonare giorni feriali dalle 14 alle 15. Ascanio Vitale - Piazza Mondragone, 9 - 80132 Napoli - Tel. 081/40269

Vendo o cambio i seguenti programmi per C64: Easy Script, Turbo Tape, Screen Graphics, Simons' BASIC, Slalom, Stix, International Soccer, Biliardo, Pole Position, Super Pipeline, Frogger. Disponibili anche registrati con Turbo Tape. Prezzi minimi. Allegare francobollo per la risposta. Lorenzo Giannono - Via Zanelli, 19 - 25100 Brescia - Tel. 030/294505

Vendo cassetta con 20 programmi e videogiochi a sole L. 25.000 (Pac Man, Pole Position, Apocalypse, Soccer, Simons' BASIC, ecc.). Per informazioni scrivere inviando L. 400 in francobolli per la risposta a: Franco Balsamo - Via Mazzini, 7 - 97013 Comiso (RG)

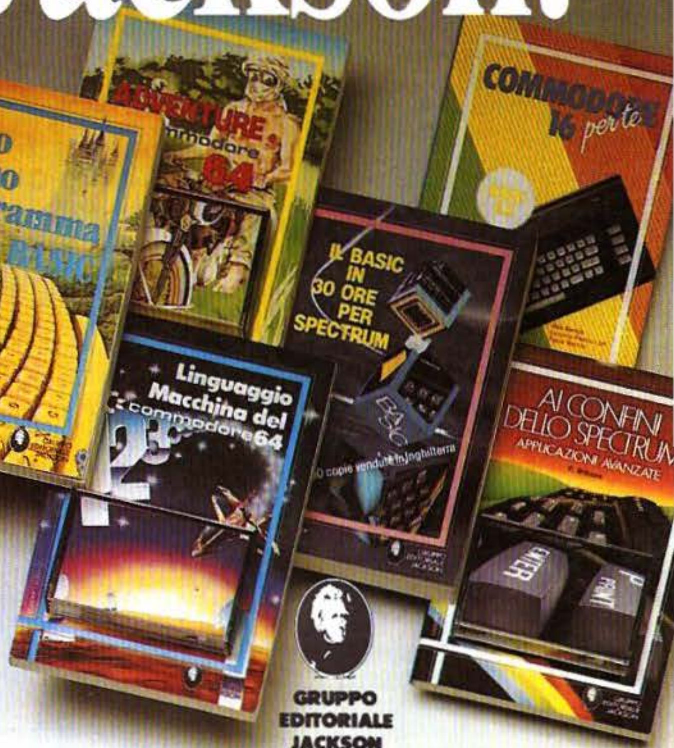
Vendo VIC 20 in ottime condizioni più 2 cartucce (Cosmic Cruncher e Visible Solar System), libro "Alla scoperta del VIC 20", manuale, cassetta giochi ed utilities, trasformatore. Vendo anche separatamente. Telefonare ore pasti a: Giuseppe Mazzacane - Via G. Fanelli, 226/40 - 70100 Bari - Tel. 080/481614

Vendo software da bar per il CBM 64 (Quix, Donkey Kong, Pole Position, ecc.) a prezzi stracciati: 10 giochi L. 20.000. Scrivete o telefonate. Davide Cutrino - Via Ciconi, 26 - 35100 Padova - Tel. 049/660006

CBM 16: per primi in Italia disponiamo di un package di programmi gestionali e videogames. La cassetta o il floppy, comprese istruzioni in italiano (supporti compresi), L. 80.000. Per ordini: La Scuola per l'Informatica - Casella Postale 17011 - 20170 Milano

Vendo programmi per C64. Vasta scelta di video games ed utility. Disponibile cassetta o disco. Gradita lista per evitare eventuali doppioni. Rispondo a tutti. Claudio Luciani - Via Gagarin - 63020 Piane di Falerone (AP)

Novità Jackson.



GRUPPO
EDITORIALE
JACKSON

David Lawrence

LINGUAGGIO MACCHINA DEL COMMODORE 64

Il libro apre nuovi orizzonti a tutti coloro che sono interessati alla programmazione in linguaggio macchina del COMMODORE 64.

Con cassetta
Cod. 572D Pag. 208 Lire 29.000

Clive Prigmore

IL BASIC IN 30 ORE PER SPECTRUM

Questo semplice corso di autoistruzione insegna a programmare, e un programma ha sempre bisogno di due ingredienti, un linguaggio e una struttura: dunque questo libro non insegna solo il BASIC, ma anche come si organizza correttamente un buon programma.

Cod. 501B Pag. 360 Lire 40.000

Rodnay Zaks

IL TUO PRIMO PROGRAMMA IN BASIC

La diffusione del BASIC per la sua semplicità e quasi "naturalezza" di programmazione fa sì che una cultura generale sull'informatica e la sua applicazione non può prescindere da una conoscenza di base di questo linguaggio. Questo lo scopo del libro: permettere anche a chi ha soltanto una cultura di base, di capire che cos'è il BASIC e come si usa.

Cod. 507B Pag. 216 Lire 19.500

Czes Kosniowski

MATEMATICA E COMMODORE 64

Tutte le funzioni matematiche disponibili sul C64 sono qui descritte, ed il loro uso è illustrato con programmi che possono essere utilizzati dal lettore all'interno dei suoi, per particolari applicazioni.
Il libro contiene anche informazioni e programmi su altri argomenti, come i codici e la crittografia, i numeri casuali, le serie, la trigonometria, i numeri primi e l'analisi statistica dei dati.

Con cassetta
Cod. 570D Pag. 160 Lire 24.000

P. Williams

AI CONFINI DELLO SPECTRUM
Applicazioni avanzate

Un esame attento dei listati consentirà al lettore di apprendere i "segreti" della programmazione strutturata e migliorare notevolmente le proprie capacità di programmatore.
I programmi presentati vanno dagli arcade più famosi, tra cui il celebre "Spectrum Invaders" ai programmi di utilità più interessanti, dai giochi d'azzardo ai programmi didattici, dai programmi funzionali a quelli di matematica e di giochi di strategia.

Con cassetta
Cod. 414B Pag. 180 Lire 28.000

Mike Grace

ADVENTURE E COMMODORE 64

Un manuale per ideare e utilizzare programmi di Adventure basati esclusivamente sul testo.
Una progettazione modulare del programma rende più facile la comprensione della struttura. L'abilità nella programmazione avrà modo di emergere durante la costruzione graduale dell'Adventure, per modificare il programma dimostrativo, o addirittura per costruirne uno nuovo.

Con cassetta
Cod. 571D Pag. 240 Lire 35.000

Rita Bonelli

COMMODORE 16 PER TE: BASIC 3.5

È un libro di introduzione al BASIC C16 con il classico taglio didattico Bonelli. Adatto per la Scuola media inferiore e per chi non conosce l'informatica.
La cassetta allegata al libro contiene diverse lezioni, una per ogni capitolo, che devono essere lette prima del capitolo relativo.

Con cassetta
Cod. 413B Pag. 296 Lire 35.000

La biblioteca che fa testo.

ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON – Divisione Libri - Via Rosellini, 12 - 20124 Milano

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
		Totale	

☐ Pagherò contrassegno al postino il prezzo indicato più L. 3.000 **per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ____________
n° ____________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato
☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome ____________
Cognome ____________
Via ____________
Cap ______ Città ______ Prov. ______
Data ______ Firma ______

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|

ORDINE MINIMO L. 50.000

SUPERCOMMODORE — LIBRI

CEDOLA DI ORDINAZIONE - LIBRI
da compilare e spedire in busta chiusa a
J.soft - Viale Restelli, 5 - 20124 Milano - Tel. 6888228

Data base:
l'utile e il dilettevole cod. AJAJ009 L. 13.500
Il libro dei giochi
per Commodore 16 e Plus/4 cod. ASOC012 L. 21.600

Ordino i seguenti libri per un importo totale di L. .... + L. 2.000
come contributo fisso per spese di spedizione

Cod. .... Cod. .... Cod. ....

Cod. .... Cod. .... Cod. ....

☐ Contanti allegati ☐ Assegno allegato n° ....

....

☐ Ho spedito l'importo a mezzo vaglia postale

☐ Ho versato l'importo sul ccp. n° 19445204 intestato a J.soft - Milano

☐ Pagherò in contrassegno al postino al ricevimento dei volumi (valido solo per i soci in Italia)

Nome ....

Cognome ....

Via ....

Città .... C.A.P. .... Prov. ....

Se richiesta fattura:
Cod. F. e P. Iva ....

Data ....

Firma ....

Per i residenti all'estero: pagamento anticipato (vaglia o versamento su ns. ccp)

SUPERCOMMODORE è bello, però... (ovvero suggerimenti, idee, critiche, richieste e tutto ciò che vi passa per la testa).

....

....

....

....

....

....

....

Nome ....

Cognome ....

Via .... C.A.P. ....

Città .... Tel. ....

SUPERCOMMODORE

La rubrica INPUT/OUTPUT è gratuita ed aperta a tutti i lettori. Chi desidera comprare, vendere o cambiare hardware o software può inviare il tagliando a J.soft - Viale Restelli, 5 - 20124 MILANO

INPUT/OUTPUT

☐ COMPRO ☐ VENDO ☐ CAMBIO

☐ VIC 20 ☐ C 16 ☐ C64 ☐ PERIF. ☐ SOFTWARE

....

....

....

....

....

....

Nome .... Cognome ....

Via .... C.A.P. ....

Città .... Tel. ....

Dietro un bambino maltrattato c'è sempre una famiglia in crisi.

È impossibile stabilire con esattezza quanti bambini in Italia subiscono violenze. Tanti, purtroppo. Ma una cosa è certa: la violenza che oggi un bambino subisce e impara, da grande la ritorcerà contro la società. Bisogna intervenire, e subito: agendo sulle cause. Intervenendo sulla crisi familiare che è sempre alla base dei maltrattamenti a un bambino.

"Vorrei contribuire anch'io".
Per sostenere il C.A.F. vi invio

☐ *L. 10.000* ☐ *L. 25.000* ☐ *L. 50.000*
☐ *L. 100.000* ☐ *L. 500.000* ☐ *L.*

☐ *Tramite assegno bancario intestato C.A.F. Via V. E. Orlando 15, 20142 Milano, che vi invio, unito a questo tagliando.*

☐ *Tramite versamento sul C.C.P. n° 22349203 intestato C.A.F. Via V. E. Orlando 15, 20142 Milano.*

Nome e Cognome ....

Indirizzo ....

CAP .... *Città* ....

Provincia ....

Attendo: ricevuta e tesserino di "Sostenitore" del C.A.F.

JAR

Segnaliamo ai Servizi Sociali del Territorio i casi di maltrattamento.

La prima cosa da fare: segnalare ogni caso di maltrattamento di cui siamo a conoscenza o di cui abbiamo il fondato sospetto. Telefonate ai Servizi Sociali del Territorio dal lunedì al venerdì, dalle ore 9 alle ore 12 (troverete i numeri di telefono su Tuttocittà), oppure al Tribunale per i Minorenni (chiedendo del giudice di turno) dal lunedì al sabato dalle ore 9 alle ore 13, telefono 4988001/2/3: oppure al C.A.F., a questi numeri di telefono 02/8265051/2, che rispondono 24 ore su 24, 7 giorni su 7.

Sosteniamo il C.A.F.

Il C.A.F. è il primo Centro di Aiuto al Bambino Maltrattato e alla Famiglia in Crisi, sorto in Italia con lo scopo di intervenire nei casi di maltrattamento.

Oggi, nella sua nuova sede più grande e attrezzata, accoglieprovvisoriamente i bambini maltrattati, contemporaneamente svolge un'azione terapeutica sulla famiglia per ripristinare l'equilibrio e reintegrare il bambino nel suo ambiente.

Il C.A.F. è un'istituzione privata che opera in collaborazione con i Servizi Sociali, con i supporti messi gratuitamente a disposizione dal Comune di Milano, e con i contributi finanziari volontari dei cittadini. Contribuisci anche tu, utilizzando il tagliando. È un gesto necessario, per i bambini maltrattati, per te che vuoi fare qualcosa di concreto contro la violenza.

UN BAMBINO MALTRATTATO OGGI SARÀ UN UOMO VIOLENTO DOMANI. FACCIAMO QUALCOSA PRIMA.

Spazio gratuito offerto da J.*soft*

C.A.F.



CENTRO DI AIUTO AL BAMBINO MALTRATTATO E ALLA FAMIGLIA IN CRISI

Associati Globe





32 BIT | 32 BIT

16 BIT | 16 BIT

8 BIT | 8 BIT

SINCLAIR QL: AL VERTICE DELLA NUOVA GENERAZIONE

Sinclair QL rivoluziona il mondo dei computer, perché combina le dimensioni di un home con la potenza e le capacità di un mini.
QL è l'unico computer, nella sua fascia, ad impiegare il microprocessore a 32 bit, quando gli altri si fermano a 8 oppure 16.
La sua portentosa memoria è di 128 KRAM espandibile a 640.
I quattro programmi applicativi, già incorporati, sono immediatamente utilizzabili e superano, in qualità, il software dei microcomputer esistenti.

Ha la possibilità di multitask e può essere inserito in reti di comunicazione.
Grazie ai due microdrive e al software incorporati, Sinclair QL, nella sua confezione originale, è già pronto per l'uso: basta collegarlo ad un video.
E pensare che tutta questa tecnologia pesa meno di due chili e trova spazio in una normale 24 ore.
Un computer così non poteva che essere Sinclair.

sinclair

Distribuzione esclusiva: GBC Divisione Rebit.

Tutti i prodotti Sinclair, distribuiti da GBC Divisione Rebit, sono corredati da regolare certificato di garanzia italiana.